1. Tổng quan về Visual Basic 6.0
2. Biểu mẫu và một số điều khiển thông dụng
3. Lập trình cấu trúc trong Visual Basic
4. Các kiểu dữ liệu có cấu trúc
5. Thiết kế biểu mẫu dùng các điều khiển
6. Lập trình xử lý giao diện & đồ họa
7. Tập tin
8. Khái niệm cơ bản về cơ sở dữ liệu
9. Các đối tượng truy cập dữ liệu
10. ODBC vả các đối tượng dữ liệu từ xa
11. Đối tượng dữ liệu ActiveX
12. Môi trường dữ liệu
13. Thiết lập báo cáo

Tổng quan về Visual Basic 6.0

Mục tiêu: Chương này giới thiệu về môi trường phát triển tích hợp (IDE) Microsoft Visual Basic 6.0; cũng như giúp sinh viên có cái nhìn tổng quan về Visual Basic.

Học xong chương này, sinh viên phải nắm bắt được các vấn đề sau:

- Sử dụng môi trường phát triển tích hợp VB 6.0 để phát triển ứng dụng.

- Cách tạo dự án mới (New Project) trong VB 6.0.

Kiến thức có liên quan:

- Sử dụng hệ điều hành Windows.

Tài liệu tham khảo:

- Visual Basic 6 Certification Exam Guide - Chapter 1, Page 1 - Dan Mezick & Scot Hillier - McGraw-Hill - 1998.

Giới thiệu về Visual Basic 6.0

Visual Basic 6.0 (VB6) là một phiên bản của bộ công cụ lập trình Visual Basic (VB), cho phép người dùng tiếp cận nhanh cách thức lập trình trên môi trường Windows. Những ai đã từng quen thuộc với VB thì tìm thấy ở VB6 những tính năng trợ giúp mới và các công cụ lập trình hiệu quả. Người dùng mới làm quen với VB cũng có thể làm chủ VB6 một cách dễ dàng.

Với VB6, chúng ta có thể :

- Khai thác thế mạnh của các điều khiển mở rộng.
- Làm việc với các điều khiển mới (ngày tháng với điều khiển MonthView và DataTimePicker, các thanh công cụ có thể di chuyển được CoolBar, sử dụng đồ họa với ImageCombo, thanh cuộn FlatScrollBar,…).
- Làm việc với các tính năng ngôn ngữ mới.
- Làm việc với DHTML.

- Làm việc với cơ sở dữ liệu.
- Các bổ sung về lập trình hướng đối tượng.

Cài đặt Visual Basic 6.0

Sử dụng chương trình Setup, người dùng có thể cài đặt VB6 lên máy tính của mình. Chương trình Setup này còn cài đặt các tập tin cần thiết để xem tài liệu trên đĩa CD MSDN (Microsoft Developer Network). Nếu cần, người dùng có thể cài đặt riêng phần tài liệu và ví dụ mẫu của Visual Basic lên máy tính.

Để cài đặt VB6, người dùng nên kiểm tra máy tính của mình đảm bảo được cấu hình tối thiểu. Các yêu cầu hệ thống tối thiểu :

- Microsoft Windows 95 trở lên hoặc là Microsoft Windows NT Workstation 4.0 trở lên.

- Tốc độ CPU 66 MHz trở lên.

- Màn hình VGA hoặc màn hình có độ phân giải cao được hỗ trợ bởi Microsoft Windows.

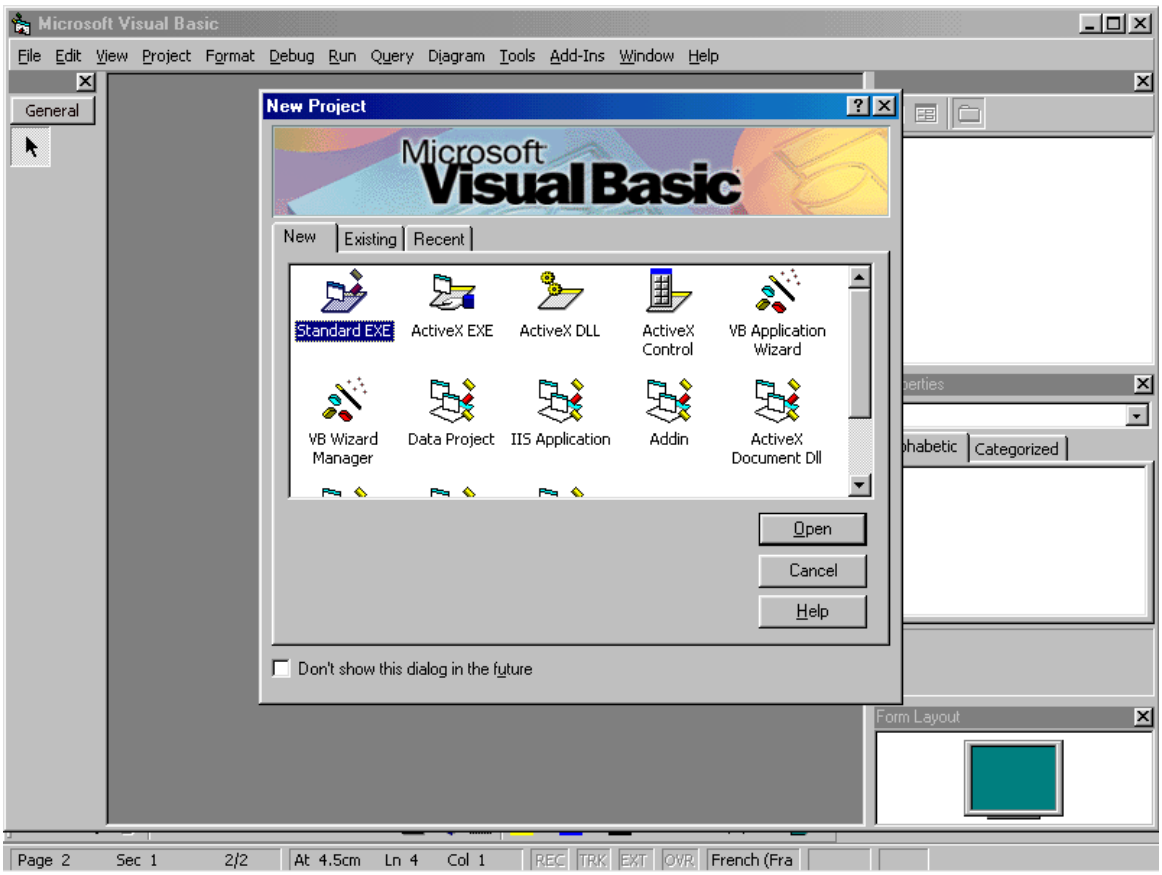
- 16 MB RAM cho Microsoft Windows 95 hoặc 32MB RAM cho Microsoft Windows NT Workstation.

Làm quen với VB6

Bắt đầu một dự án mới với VB6

Từ menu Start chọn Programs, Microsoft Visual Basic 6.0. Khi đó bạn sẽ thấy màn hình đầu tiên như hình I.1 dưới đây.

Hình I.1 Cửa sổ khi kích hoạt VB6



[missing_resource: .png]

Chọn

Ở đây, người dùng có thể chọn tạo mới một dự án thực thi được bằng cách chọn Standard EXE rồi nhấp Open (Hình I.2).

Tiếp theo là cửa sổ làm việc chính của VB6, gọi tắt là IDE (Integrated Development Environment) sẽ được giới thiệu chi tiết trong phần sau.

Tìm hiểu các thành phần của IDE

IDE là tên tắt của môi trường phát triển tích hợp (Integrated Development Environment), đây là nơi tạo ra các chương trình Visual Basic.

IDE của Visual Basic là nơi tập trung các menu, thanh công cụ và cửa sổ để tạo ra chương trình. Mỗi một thành phần của IDE có các tính năng ảnh hưởng đến các hoạt động lập trình khác nhau.

[missing_resource: .png]

Hình I.3 Cửa sổ IDE của VB6

Thanh menu cho phép bạn tác động cũng như quản lý trực tiếp trên toàn bộ ứng dụng. Bên cạnh đó thanh công cụ cho phép truy cập các chức năng của thanh menu thông qua các nút trên thanh công cụ.

Các biểu mẫu (Form) - khối xây dựng chương trình chính của VB - xuất hiện trong cửa sổ Form. Hộp công cụ để thêm các điều khiển vào các biểu mẫu của đề án. Cửa sổ Project Explorer hiển thị các đề án khác nhau mà người dùng đang làm cũng như các phần của đề án. Người dùng duyệt và cài đặt các thuộc tính của điều khiển, biểu mẫu và module trong cửa sổ Properties. Sau cùng, người dùng sẽ xem xét và bố trí một hoặc nhiều biểu mẫu trên màn hình thông qua cửa sổ Form Layout.

Sử dụng thanh công cụ trong IDE của VB

Thanh công cụ là tập hợp các nút bấm mang biểu tượng thường đặt dưới thanh menu. Các nút này đảm nhận các chức năng thông dụng của thanh menu (New, Open, Save ...).
[missing_resource: .png]

Hình I.4 Thanh công cụ ở dạng standard

Hơn nữa, người dùng có thể kéo rê thanh công cụ trên IDE đến vị trí bất kỳ nào đó thuận tiện cho việc sử dụng.
[missing_resource: .png]

Hình I.5 Popup menu thêm, xóa công cụNgười dùng có thể thêm hay xóa thanh công cụ trên IDE:

* Chọn Toolbars từ menu View hoặc ấn chuột phải vào điểm bất kỳ nào trên thanh menu, một popup menu bật ra.

* Chọn loại thanh công cụ mà ta muốn thêm vào hoặc xóa đi. Nếu có đánh dấu check ở bên trái thì loại công cụ đó đang được chọn.

Sử dụng thanh công cụ gỡ rối (debug)
[missing_resource: .png]

Hình I.6 Thanh công cụ gỡ rốiVới thanh công cụ gỡ rối, người dùng có thể thực thi, tạm ngưng hoặc dừng một đề án. Với thanh công cụ Debug, người dùng có thể kiểm tra chương trình và giải quyết các lỗi có thể xảy ra. Khi gỡ rối chương trình, người dùng có thể chạy từng dòng lệnh, kiểm tra giá trị các biến, dừng chương trình tại một điểm nào đó hoặc với một điều kiện nào đó.

Sử dụng thanh công cụ Edit
[missing_resource: .png]

Hình I.7 Thanh công cụ EditThanh công cụ Edit được dùng để viết chương trình trong cửa sổ Code, thanh công cụ Edit có đầy đủ các tính năng của menu Edit. Ngoài ra người sử dụng có thể sử dụng chức năng viết chương trình tự động như là Quick Info.

Thanh công cụ Edit của VB6 có tính năng lý thú đó là tự hoàn tất các từ khóa. Tính năng này rất hữu dụng giúp cho người dùng tránh các lỗi mắc phải do gõ sai từ khóa.

Sử dụng thanh công cụ Form Editor

[missing_resource: .png]

Hình I.8 Thanh công cụ thiết kế biểu mẫu

Thanh công cụ Form Editor có chức năng giống như menu Format dùng để di chuyển và sắp xếp các điều khiển trên biểu mẫu.

Trong quá trình thiết kế biểu mẫu, đôi khi chúng ta phải sử dụng thuộc tính ZOrder để cho phép một điều khiển có thể thay thế một điều khiển khác hay không hoặc là xuất hiện bên trên một điều khiển khác hay không.

Sử dụng hộp công cụ (Toolbox)

Hộp công cụ là nơi chứa các điều khiển được dùng trong quá trình thiết kế biểu mẫu. Các điều khiển được chia làm hai loại: Điều khiển có sẵn trong VB và các điều khiển được chứa trong tập tin với phần mở rộng là .OCX.

Đối với các điều khiển có sẵn trong VB thì ta không thể gỡ bỏ khỏi hộp công cụ, trong khi đó đối với điều khiển nằm ngoài ta có thêm hoặc xóa bỏ khỏi hộp công cụ.

Một điều khiển có thể được đưa vào biểu mẫu bằng cách chọn điều khiển đó và đưa vào biểu mẫu. Chúng ta sẽ trở lại phần này trong chương tiếp theo khi thiết kế các biểu mẫu.

Hình I.9 Hộp công cụ của Visual Basic

Quản lý ứng dụng với Project Explorer

Project Explorer trong VB6 giúp quản lý và định hướng nhiều đề án.VB cho phép nhóm nhiều đề án trong cùng một nhóm. Người dùng có thể lưu

tập hợp các đề án trong VB thành một tập tin nhóm đề án với phần mở rộng .vbp.
[missing_resource: .png]

Hình I.10 Cửa sổ Project ExplorerProject Explorer có cấu trúc cây phân cấp như cây thư mục trong cửa sổ Explorer của hệ điều hành. Các đề án có thể được coi là gốc của cây, các thành phần của đề án như biểu mẫu, module ... là các nút của cây. Khi muốn làm việc với thành phần nào thì ta có thể nhấn đúp lên thành phần đó trên cửa sổ Project Explorer để vào cửa sổ viết code cho thành phần đó.
[missing_resource: .png]

Hình I.11 Cửa sổ PropertiesKhi làm việc với một dự án lớn, chúng ta sẽ thấy Project Explorer cực kỳ hữu ích cho việc tổ chức và quản lý một dự án lớn.

Cửa sổ Properties

Mỗi một thành phần, điều khiển điều có nhiều thuộc tính. Mỗi một thuộc tính lại có một hoặc nhiều giá trị.

Cửa sổ Properties cho phép người dùng xem, sửa đổi giá trị các thuộc tính của điều khiển nhằm giúp điều khiển hoạt động theo đúng ý đồ của người sử dụng.

Cửa sổ Form Layout

Đây chính là cửa sổ trình bày biểu mẫu cho phép định vị trí của một hoặc nhiều biểu mẫu trên màn hình khi chương trình ứng dụng được thi hành.

Ta ñònh vò moät bieåu maãu treân maøn hình baèng caùch duøng chuoät di chuyeån bieåu maãu trong cöûa soå Form Layout.
[missing_resource: .png]

Hình I.12 Cửa sổ Form LayoutSử dụng cửa sổ Form Layout không đơn giản như các cửa sổ khác vì nó không được kích hoạt sẵn, người dùng

cần phải chạy ứng dụng sau đó mới có thể bố trí được các biểu mẫu thông qua Form Layout.

Nếu ta không định vị các biểu mẫu thì vị trí của biểu mẫu trên màn hình lúc thiết kế cũng là vị trí khởi động của biểu mẫu khi thực thi.

Biên dịch đề án thành tập tin thực thi

Sau khi đề án đã hoàn thành, người dùng có thể biên dịch thành tập tin thực thi được. Cách tiến hành như sau:

- Trước tiên ta cần chỉ cho VB6 biết phần chương trình nào sẽ được thực thi trước bằng cách chọn Project Properties từ menu Project. Chọn tab General, chú ý phần Startup Object, đây là nơi quy định điểm khởi đầu của chương trình sau khi biên dịch kết thúc.

- Từ menu File, chọn Make ... EXE... Một hộp thoại xuất hiện cho phép bạn nhập vào tên của tập tin thực thi. Bạn chỉ cần gõ tên tập tin, VB sẽ tự động thêm phần mở rộng .EXE.

- Nhấn vào nút Options để mở hộp thoại Project Properties và điền tên của ứng dụng vào ô Title, ta có thể ghi chú thông tin cho từng phiên bản trong phần Version Information. Ta có thể chọn Auto Increment để VB tự động tăng số Revision mỗi lần ta tạo lại tập tin EXE cho dự án.

- Cuối cùng, nhấn OK để trở về hộp thoại Make Project.

Biểu mẫu và một số điều khiển thông dụng
Mục tiêu: Chương này giới thiệu về một số điều khiển cơ bản để tạo nên giao diện cho các ứng dụng cũng như một số khái niệm trong lập trình với VB; những yêu cầu tối thiểu cần có trong việc "lập trình sự kiện" với VB.

Học xong chương này, sinh viên phải nắm bắt được các vấn đề sau:

- Khái niệm về điều khiển, thuộc tính, phương thức, sự kiện.

- Quy tắc đặt tên danh biểu trong VB.

- Sử dụng biểu mẫu trong thiết kế giao diện.

- Sử dụng điều khiển ô nhập liệu, nút nhấn, nhãn, khung.

Kiến thức có liên quan:

- Cách thức sử dụng môi trường phát triển VB.

Tài liệu tham khảo:

- Microsoft Visual Basic 6.0 và Lập trình Cơ sở dữ liệu - Chương 2, trang 26; Chương 3, trang 29 - Nguyễn Thị Ngọc Mai (chủ biên), Nhà xuất bản Giáo dục - 2000.

Các khái niệm

- Điều khiển: Các thành phần có sẵn để người lập trình tạo giao diện tương tác với người dùng.

Mỗi điều khiển thực chất là một đối tượng, do vậy nó sẽ có một số điểm đặc trưng cho đối tượng, chẳng hạn như các thuộc tính, các phương thức & các sự kiện.

- Thuộc tính: Các đặc trưng của một điều khiển tạo nên dáng vẻ của điều khiển đó.

- Phương thức: Các điều khiển có thể thực thi một số tác vụ nào đó, các tác vụ này được định nghĩa sẵn bên trong các phương thức (còn gọi là chương trình con: hàm & thủ tục), người lập trình có thể gọi thực thi các phương thức này nếu cần.
- Sự kiện: là hành động của người dùng tác động lên ứng dụng đang thực thi.

Thí dụ:- Nhấn phím bất kỳ trên bàn phím.

- Nhấp chuột.

Các thành phần giao diện có khả năng đáp ứng lại sự kiện. Chẳng hạn khi chúng ta nhấp chuột vào button, lúc đó button nhận biết được sự kiện này; hay như textbox nhận biết được sự kiện bàn phím tác động lên nó.

Một ứng dụng trên Windows thường được thực hiện nhờ vào việc đáp ứng lại các sự kiện của người dùng.

- Lập trình sự kiện:

Các thành phần giao diện có khả năng nhận biết được các sự kiện từ phía người dùng. Tuy nhiên khả năng đáp ứng lại các sự kiện được thực hiện bởi người lập trình.

Khi một thành phần giao diện được sử dụng, người lập trình phải xác định chính xác hành động của thành phần giao diện đó để đáp ứng lại một sự kiện cụ thể. Lúc đó người lập trình phải viết đoạn mã lệnh mà đoạn mã lệnh này sẽ được thực thi khi sự kiện xảy ra.

Chẳng hạn, trong ứng dụng Paint của Windows; khi người sử dụng nhấp chuột vào nút vẽ hình elip sau đó dùng chuột vẽ nó trên cửa sổ vẽ, một hình elip được vẽ ra.

Trong lập trình sự kiện, một ứng dụng được xây dựng là một chuỗi các đáp ứng lại sự kiện. Tất cả các hành động của ứng dụng là đáp ứng lại các sự kiện. Do vậy người lập trình cần phải xác định các hành động cần

thiết của ứng dụng; phân loại chúng; sau đó viết các đoạn mã lệnh tương ứng.

Thí dụ về đáp ứng lại sự kiện:
[missing_resource: .png]

Mã lệnh- Mã lệnh cho sự kiện Click của Ghi đĩa.---------- Mã lệnh cho sự kiện Click của In giấy----------Hình II.1: Thí dụ về đáp ứng sự kiện

- Khi người dùng không tác động vào ứng dụng, ứng dụng không làm gì cả.

- Khi người dùng nhập dữ liệu vào các ô nhập Họ và tên, Địa chỉ; sự kiện bàn phím xảy ra trên các ô nhập. Tuy nhiên, ứng dụng vẫn không làm gì cả vì không có đoạn mã lệnh nào đáp ứng các sự kiện này.

- Khi người dùng nhấp nút chọn Ghi đĩa, ứng dụng tìm kiếm trong mã lệnh của mình thấy có đoạn mã lệnh đáp ứng lại sự kiện này; lúc đó đoạn mã lệnh được thực thi.

- Tương tự như vậy đối với nút chọn In giấy.

- Cách xác lập các thuộc tính & các phương thức trong chương trình

<Thuộc tính Name của điều khiển>.<Tên thuộc tính>

<Thuộc tính Name của điều khiển>.<Tên phương thức>[(<Các tham số>)]

- Tên điều khiển (thuộc tính Name)

Đây là thuộc tính xác định tên của điều khiển trong ứng dụng. Tên này được đặt theo quy tắc:

- Tên có thể dài từ 1 - 40 ký tự.
- Tên phải bắt đầu với ký tự chữ, có thể chữ hoa hay thường.

- Sau ký tự đầu tiên, tên có thể chứa ký tự, số hay dấu gạch dưới.

Ví dụ: Num, StudentCode, Class12A2 là những tên hợp lệ. 345, 7yu là những tên không hợp lệ.

Biểu mẫu (Form)

Khái niệm:

Chương trình ứng dụng giao tiếp với người dùng thông qua các biểu mẫu (hay còn gọi là cửa sổ, xuất phát từ chữ Form hay Windows); các điều khiển (Control) được đặt lên bên trên giúp cho biểu mẫu thực hiện được công việc đó.

Biểu mẫu là các cửa số được lập trình nhằm hiển thị dữ liệu và nhận thông tin từ phía người dùng.

Thuộc tính

- Name: thuộc tính này như là một định danh nhằm xác định tên của biểu mẫu là gì? Ta sẽ sử dụng thuộc tính này để truy xuất đến các thuộc tính khác cùng với phương thức có thể thao tác được trên biểu mẫu.
- Caption: chuỗi hiển thị trên thanh tiêu đề của biểu mẫu.
- Icon: hình icon được dùng trong thanh tiêu đề của biểu mẫu, nhất là khi biểu mẫu thu nhỏ lại.
- WindowState: xác định biểu mẫu sẽ có kích thước bình thường (Normal=0), hay Minimized (=1), Maximized =(2).
- Font: xác lập Font cho biểu mẫu. Thuộc tính này sẽ được các điều khiển nằm trên nó thừa kế. Tức là khi ta đặt một điều khiển lên biểu mẫu, thuộc tính Font của điều khiển ấy sẽ tự động trở nên giống y của biểu mẫu.
- BorderStyle: xác định dạng của biểu mẫu.

Phương thức

- Move: di chuyển biểu mẫu đến tọa độ X,Y: Move X, Y.

Sự kiện

- Form_Initialize: Sự kiện này xảy ra trước nhất và chỉ một lần thôi khi ta tạo ra thể hiện đầu tiên của biểu mẫu. Ta dùng sự kiện Form_Initialize để thực hiện những gì cần phải làm chung cho tất cả các thể hiện của biểu mẫu này.
- Form_Load: Sự kiện này xảy ra mỗi lần ta gọi thể hiện một biểu mẫu. Nếu ta chỉ dùng một thể hiện duy nhất của một biểu mẫu trong chương trình thì Form_Load coi như tương đương với Form_Initialize.

Ta dùng sự kiện Form_Load để khởi tạo các biến, điều khiển cho các thể hiện của biểu mẫu này.

- Form_Activate: Mỗi lần một biểu mẫu được kích hoạt (active) thì một sự kiện Activate phát sinh. Ta thường dùng sự kiện này để cập nhật lại giá trị các điều khiển trên biểu mẫu.
- Form_QueryUnload: Khi người sử dụng chương trình nhấp chuột vào nút X phía trên bên phải để đóng biểu mẫu thì một sự kiện QueryUnload được sinh ra. Đoạn chương trình con dưới đây mô tả thủ tục xử lý sự kiện QueryUnload.

Private Sub Form_QueryUnload(Cancel As Integer, _

UnloadMode As Integer)

End Sub

Sự kiện này cho ta khả năng hủy bỏ hành động đóng biểu mẫu bằng cách đặt lại Cancel là 1.

- Form_Resize: Sự kiện này xảy ra mỗi khi biểu mẫu thay đổi kích thước.

Nhãn (Label)

Khái niệm:

Nhãn là điều khiển dạng đồ họa cho phép người sử dụng hiển thị chuỗi ký tự trên biểu mẫu nhưng họ không thể thay đổi chuỗi ký tự đó một cách trực tiếp.

Biểu tượng (shortcut) trên hộp công cụ:

Thuộc tính:

- Name: Đây là một tên xác định một định danh, người lập trình có thể thay đổi tên này theo cách của mình để tiện sử dụng.
- Caption: Thuộc tính quy định chuỗi ký tự hiển thị khi ta tạo một điều khiển nhãn. Khi ta tạo mới một điều khiển thì thuộc tính Caption có giá trị mặc nhiên là “Label…”.

Ví dụ: Ta muốn tạo một nhãn là “Chào mừng bạn đến với Visual Basic”, ta thay đổi giá trị của thuộc tính Caption thành “Chào mừng bạn đến với Visual Basic”.

Ta có thể thay đổi giá trị của thuôc tính Caption tại thời điểm ứng dụng đang chạy nhờ vào đoạn mã lệnh đơn giản như sau:

L1.Caption = "Đã đổi giá trị Caption" với L1 là tên của điều khiển nhãn mà ta muốn đổi.

- Font, Fore Color: Quy định kiểu chữ, kích thước, màu hiển thị.
- BackStyle, BackColor: BackStyle quy định là nhãn trong suốt hay không. BackColor quy định màu nền của nhãn trong trường hợp không trong suốt.

Phương thức:

- Move: di chuyển nhãn đến tọa độ X,Y: Move X, Y.

Sự kiện:

- Change: Xảy ra mỗi khi nhãn thay đổi giá trị.
- Click: Mỗi khi nhãn được chuột nhấp lên, sự kiện này xảy ra.
- DblClick: Xảy ra khi người sử dụng nhấp đúp chuột lên điều khiển nhãn.

Khung (Frame)

Khái niệm:

SORRY, THIS MEDIA TYPE IS NOT SUPPORTED. Khung là một điều khiển dùng trong việc bố trí giao diện của biểu mẫu một cách trong sáng và rõ nét. Thông thường các điều khiển cùng phục vụ cho một công việc nào đó sẽ được đặt trong một khung nhằm làm nổi bật vai trò của chúng.

Biểu tượng (shortcut) trên hộp công cụ:

Khi chúng ta tạo mới một khung để chứa các điều khiển khác, ta có hai cách thực hiện:

- Tạo khung chứa trước, sau đó đưa các điều khiển vào trong khung chứa. Đây là cách đơn giản nhất.

- Tạo khung chứa sau khi đã tạo mới các điều khiển, khi đó khung chứa sẽ che mất các điều khiển, vì vậy ta cần phải đưa khung chứa ra sau các điều khiển bằng cách nhấp chuột phải và chọn Send to Back. Nhưng đối với cách này, các điều khiển khác không nằm trên khung chứa. Do vậy ta có thể giải quyết bằng cách cắt (Cut) các điều khiển này đi, sau đó dán (Paste) vào trong khung chứa.

Hình II.2 Ví dụ về khung chứa (Frame)

[missing_resource: .png]

Thuộc tính: Khung cũng có các thuộc tính thông dụng như của điều khiển nhãn chẳng hạn như: Name, Caption,...

Phương thức:

- Move: di chuyển khung đến tọa độ X,Y: Move X, Y.

Sự kiện:

- Click, DblClick: xảy ra khi khung nhận được một thao tác nhấp (nhấp đúp) chuột.

Nút lệnh (Command Button)

Khái niệm:

Nút lệnh là một điều khiển dùng để bắt đầu, ngắt hoặc kết thúc một quá trình. Khi nút lệnh được chọn thì nó trông như được nhấn xuống, do đó nút lệnh còn được gọi là nút nhấn (Push Button). Người sử dụng luôn có thể chọn một nút lệnh nào đó bằng cách nhấn chuột trên nút lệnh đó.

Biểu tượng (shortcut) trên hộp công cụ:

Thuộc tính:

- Name: sử dụng như một định danh nhằm xác định tên của nút lệnh.
- Caption: Dùng để hiển thị một chuỗi nào đó trên nút lệnh.
- Default: Nếu giá trị của thuộc tính này là True thì ta có thể chọn nút lệnh bằng cách nhấn phím Enter.
- Cancel: Nếu giá trị của thuộc tính này là True thì ta có thể chọn nút lệnh nào đó bằng cách nhấn phím ESC.
- Enabled: Trong một biểu mẫu, có thể có nhiều nút lệnh để thực hiện nhiều công việc khác nhau và tại một thời điểm nào đó ta chỉ được phép thực hiện một số công việc. Nếu giá trị thuộc tính Enabled là False thì nút lệnh đó không có tác dụng. Giá trị mặc định của thuộc tính này là True. Ta có thể thay đổi giá trị của thuộc tính tại thời điểm chạy ứng dụng.
- ToolTipText: cho phép hiển thị một đoạn văn bản chú thích công dụng của nút lệnh khi người sử dụng dùng chuột rê trên nút nhấn.
- Font, Fore Color: Quy định kiểu chữ, kích thước, màu hiển thị.

Phương thức

- Move: di chuyển nút lệnh đến tọa độ X,Y: Move X, Y.

Phương thức

- Click: đây là sự kiện thường xảy ra với nút lệnh. Mỗi khi một nút lệnh được chọn, sự kiện này được kích hoạt. Do đó, người sử dụng sẽ viết mã các lệnh để đáp ứng lại sự kiện này.

Ví dụ: Tạo một biểu mẫu có một ô nhập liệu với nhãn là họ tên và một nút lệnh cho phép đưa ra câu chào người dùng đó.

Private Sub Command1_Click()

MsgBox "Chao mung ban " & Text1.Text & _

" lam quen voi Visual Basic"

End Sub

[missing_resource: .png]

[missing_resource: .png]

Click hereHình II.3 Sử dụng nút lệnh

Ô nhập liệu (TextBox)

Khái niệm:

Ô nhập liệu là một điều khiển cho phép nhận thông tin do người dùng nhập vào. Đối với ô nhập liệu ta cũng có thể dùng để hiển thị thông tin, thông tin này được đưa vào tại thời điểm thiết kế hay thậm chí ở thời điểm thực thi ứng dụng. Còn thao tác nhận thông tin do người dùng nhập vào dĩ nhiên là được thực hiện tại thời điểm chạy ứng dụng.

SORRY, THIS MEDIA TYPE IS NOT SUPPORTED. Biểu tượng (shortcut) trên hộp công cụ

Thuộc tính:

- Name: Đây là tên của ô nhập liệu, được sử dụng như một định danh.
- MaxLength: Thuộc tính quy định số ký tự tối đa có thể nhập vào ô nhập liệu. Nếu số ký tự nhập vào vượt quá số ký tự tối đa thì chỉ có đúng số ký tự tối đa được ghi nhận vào trong thuộc tính Text.
- Text: Dùng để nhập vào thông tin cần hiển thị trong Textbox tại thời điểm thiết kế hoặc nhận giá trị do người dùng nhập vào tại thời điểm chạy ứng dụng.

Ví dụ:

MsgBox Text1.Text

Đoạn mã này viết trong sự kiện Click của nút lệnh OK. Cho phép hộp thông báo hiển thị nội dung do người dùng nhập vào ô nhập liệu.

SORRY, THIS MEDIA TYPE IS NOT SUPPORTED.

Hình II.3 Ví dụ về điều khiển ô nhập liệu

- Locked: Thuộc tính cho phép người dùng thay đổi nội dung của ô nhập liệu được hay không? Thuộc tính này có thể nhận 2 giá trị True hoặc False. Nếu False thì người dùng có thể thay đổi nội dung của ô nhập liệu & mặc định thì thuộc tính này có giá trị là False.
- PasswordChar: Thuộc tính này quy định cách hiển thị thông tin do người dùng nhập vào. Chẳng hạn, nếu ta nhập vào giá trị thuộc tính này là * thì các ký tự nhập vào điều hiển thị bởi dấu * . Thuộc tính này thường được dùng trong trường hợp thông tin nhập vào cần được che giấu (Ví dụ mật khẩu đăng nhập một chương trình ứng dụng nào đó mà trong đó các người dùng khác nhau thì có các quyền khác nhau).
- Multiline: Thuộc tính quy định ô nhập liệu có được hiển thị thông tin dưới dạng nhiều hàng hay không, nếu là TRUE thì ô nhập liệu cho phép nhiều hàng.
- Font, Fore Color: Quy định kiểu chữ, kích thước, màu hiển thị.

- SelLength:Cho phép trả về hoặc đặt trước số lượng ký tự được chọn trong ô nhập liệu.
- SelStart: Trả về hoặc xác định điểm bắt đầu của chuỗi được chọn. Đây là vị trí bắt đầu chèn một chuỗi mới trong trường hợp không có đánh dấu chọn chuỗi.
- SelText: Trả về hoặc xác định chuỗi ký tự được đánh dấu chọn, chỗi trả về sẽ là rỗng nếu như không đánh dấu chọn chuỗi nào.

Ba thuộc tính SelLength, SelStart, SelText chỉ có tác dụng tại thời điểm chạy ứng dụng.

Phương thức

- Move: Di chuyển ô nhập liệu đến tọa độ X, Y: Move X, Y.
- SetFocus: Phương thức này nhằm mục đích thiết lập cho điều khiển ô nhập liệu nhận được Focus, nghĩa là nó sẵn sàng được tương tác bởi người sử dụng.

Sự kiện:

- KeyPress: xảy ra khi người sử dụng chương trình nhấn một phím. Đối với điều khiển TextBox, ta thường dùng nó để lọc (filter out) các phím không chấp nhận. Sự kiện KeyPress cho ta một mã Ascii, một số có giá trị từ 0 đến 255, của phím vừa nhấn. Trong ví dụ dưới đây, TextBox Text1 sẽ chỉ nhận biết các phím là số (0 - 9), không nhận biết các phím khác:

```
Private Sub Text1_KeyPress(KeyAscii As Integer)

If KeyAscii < 48 Or KeyAscii > 57 Then ‘ Mã Ascii của 0 là 48, của 9 là 57

KeyAscii = 0
```

End If

End Sub

- KeyDown, KeyUp: mỗi sự kiện KeyPress lại cho ta một cặp sự kiện KeyDown/KeyUp. Sự kiện KeyDown/KeyUp có 2 tham số là KeyCode và Shift. Sự kiện này cho phép ta nhận biết được các phím đặc biệt trên bàn phím. Trong ví dụ dưới đây, ta hiển thị tên các phím chức năng mà người sử dụng chương trình nhấn vào:

Private Sub Text3_KeyDown(KeyCode As Integer, Shift As Integer)

If (KeyCode >= 112) And (KeyCode <= 123) Then

MsgBox "Ban vua nhan phim chuc nang: F" & _

Trim(Str(KeyCode - 111))

End If

End Sub

Lập trình cấu trúc trong Visual Basic
Mục tiêu: Chương này giới thiệu về các cấu trúc lập trình trong VB; đây là các cấu trúc cốt lõi để xây dựng nên một chương trình VB.

Học xong chương này, sinh viên phải nắm bắt được các vấn đề sau:

- Sử dụng môi trường lập trình VB để viết mã lệnh.

- Các kiểu dữ liệu trong VB.

- Cách khai báo hằng, biến trong VB.

- Biểu thức trong VB.

- Các câu lệnh đơn cũng như các câu lệnh có cấu trúc.

- Chương trình con trong VB.

- Bẫy lỗi trong VB.

Kiến thức có liên quan:

- Cách sử dụng môi trường phát triển của VB.

Tài liệu tham khảo:

- Microsoft Visual Basic 6.0 và Lập trình Cơ sở dữ liệu - Chương 4, trang 49 - Nguyễn Thị Ngọc Mai (chủ biên), Nhà xuất bản Giáo dục - 2000.

Môi trường lập trình

Soạn thảo chương trình:

Trong Visual Basic IDE, cửa sổ mã lệnh (Code) cho phép soạn thảo chương trình. Cửa sổ này có một số chức năng nổi bật:

- Đánh dấu (Bookmarks): Chức năng này cho phép đánh dấu các dòng lệnh của chương trình trong cửa sổ mã lệnh để dễ dàng xem lại về

sau này. Để bật tắt khả năng này, chọn Bookmarks từ menu Edit, hoặc chọn từ thanh công cụ Edit.

- Các phím tắt trong cửa sổ mã lệnh:

Chöùc naêng	Phím taét
Xem cöûa soå Code	F7
Xem cöûa soå Object Browser	F2
Tìm kieám	CTRL+F
Thay theá	CTRL+H
Tìm tieáp	SHIFT+F4
Tìm ngöôïc	SHIFT+F3
Chuyeån ñeán thuû tuïc keá tieáp	CTRL+DOWN ARROW
Chuyeån ñeán thuû tuïc tröôùc ñoù	CTRL+UP ARROW
Xem ñònh nghóa	SHIFT+F2
Cuoän xuoáng moät maøn hình	CTRL+PAGE DOWN
Cuoän leân moät maøn hình	CTRL+PAGE UP
Nhaûy veà vò trí tröôùc ñoù	CTRL+SHIFT+F2
Trôû veà ñaàu cuûa moâ-ñun	CTRL+HOME

Ñeán cuoái moâ-ñun	CTRL+END

Các chức năng tự động:

- Tự động kiểm tra cú pháp (Auto Syntax Check)

Nếu chức năng này không được bật thì khi ta viết một dòng mã có chứa lỗi, VB chỉ hiển thị dòng chương trình sai với màu đỏ nhưng không kèm theo chú thích gì và tất nhiên ta có thể viết tiếp các dòng lệnh khác. Còn khi chức năng này được bật, VB sẽ cho ta biết một số thông tin về lỗi và hiển thị con trỏ ngay dòng chương trình lỗi để chờ ta sửa.

- Yêu cầu khai báo biến (Require Variable Declaration)

VB sẽ thông báo lỗi khi một biến được dùng mà không khai báo và sẽ chỉ ra vị trí của biến đó.

[missing_resource: .png]

Hình III.1: Cửa sổ Options

- Gợi nhớ mã lệnh (Code):

Khả năng Auto List Members: Tự động hiển thị danh sách các thuộc tính và phương thức của 1 điều khiển hay một đối tượng khi ta gõ vào tên của chúng. Chọn thuộc tính hay phương thức cần thao tác và nhấn phím Tab hoặc Space để đưa nó vào chương trình.

[missing_resource: .png]

Hình III.2 Cửa sổ Code với khả năng gợi nhớ Code

Kiểu dữ liệu

Khái niệm

Kiểu dữ liệu là một tập hợp các giá trị mà một biến của kiểu có thể nhận và một tập hợp các phép toán có thể áp dụng trên các giá trị đó.

Các kiểu dữ liệu cơ sở trong Visual Basic

Kiểu dữ liệu	Mô tả
Boolean	Gồm 2 giá trị: TRUE & FALSE.
Byte	Các giá trị số nguyên từ 0 – 255
Integer	Các giá trị số nguyên từ -32768 – 32767
Long	Các giá trị số nguyên từ -2147483648 – 2147483647. Kiểu dữ liệu này thường được gọi là số nguyên dài.
Single	Các giá trị số thực từ -3.402823E+38 – 3.402823E+38. Kiểu dữ liệu này còn được gọi là độ chính xác đơn.
Double	Các giá trị số thực từ -1.79769313486232E+308 - 1.79769313486232E+308. Kiểu dữ liệu này được gọi là độ chính xác kép.
Currency	Dữ liệu tiền tệ chứa các giá trị số từ -922.337.203.685.477,5808 - 922.337.203.685.477,5807.

String	Chuỗi dữ liệu từ 0 đến 65.500 ký tự hay ký số, thậm chí là các giá trị đặc biệt như ^%@. Giá trị kiểu chuỗi được đặt giữa 2 dấu ngoặc kép (“”).
Date	Dữ liệu kiểu ngày tháng, giá trị được đặt giữa cặp dấu ##. Việc định dạng hiển thị tùy thuộc vào việc thiết lập trong Control Panel.
Variant	Chứa mọi giá trị của các kiểu dữ liệu khác, kể cả mảng.

Hằng số

Khái niệm

Hằng số (Constant) là giá trị dữ liệu không thay đổi.

Khai báo hằng

[Public|Private] Const <tên hằng> [As <kiểu dữ liệu>] = <biểu thức>

Trong đó, tên hằng được đặt giống theo quy tắc đặt tên của điều khiển.

Ví dụ:

Const g = 9.8

Const Num As Integer = 4*5

Ta có thể dùng cửa sổ Object Browser để xem danh sách các hằng có sẵn của VB và VBA (Visual Basic for Application).

Trường hợp trùng tên hằng trong những thư viện khác nhau, ta có thể chỉ rõ tham chiếu hằng.

[<Libname>.][<tên mô-đun>.] <tên hằng>

Biến

Khái niệm

Biến (Variable) là vùng lưu trữ được đặt tên để chứa dữ liệu tạm thời trong quá trình tính toán, so sánh và các công việc khác.

Biến có 2 đăc điểm:

- Mỗi biến có một tên.
- Mỗi biến có thể chứa duy nhất một loại dữ liệu.

Khai báo

[Public|Private|Static|Dim] <tên biến> [As <kiểu dữ liệu>]

Trong đó, tên biến: là một tên được đặt giống quy tắc đặt tên điều khiển. Nếu cần khai báo nhiều biến trên một dòng thì mỗi khai báo cách nhau dấu phẩy (,).

Nếu khai báo biến không xác định kiểu dữ liệu thì biến đó có kiểu Variant.

Khai báo ngầm: Đây là hình thức không cần phải khai báo một biến trước khi sử dụng. Cách dùng này có vẻ thuận tiện nhưng sẽ gây một số sai sót, chẳng hạn khi ta đánh nhầm tên biến, VB sẽ hiểu đó là một biến mới dẫn đến kết quả chương trình sai mà rất khó phát hiện.

Ví dụ:

Dim Num As Long, a As Single

Dim Age As Integer

Khai báo tường minh: Để tránh rắc rối như đã nêu ở trên, ta nên quy định rằng VB sẽ báo lỗi khi gặp biến chưa được khai báo bằng dòng lệnh:

Option Explicit trong phần Declaration (khai báo) của mô-đun.

Option Explicit chỉ có tác dụng trên từng mô-đun do đó ta phải đặt dòng lệnh này trong từng mô-đun của biểu mẫu, mô-đun lớp hay mô-đun chuẩn.

Biểu thức

Khái niệm

Toán tử hay phép toán (Operator): là từ hay ký hiệu nhằm thực hiện phép tính và xử lý dữ liệu.

Toán hạng: là giá trị dữ liệu (biến, hằng…).

Biểu thức: là tập hợp các toán hạng và các toán tử kết hợp lại với nhau theo quy tắc nhất định để tính toán ra một giá trị nào đó.

Các loại phép toán

1. Các phép toán số học: Thao tác trên các giá trị có kiểu dữ liệu số.

Phép	Ý nghĩa	Kiểu của đối số	Kiểu của

toán			kết quả
-	Phép lấy số đối	Kiểu số (Integer, Single…)	Như kiểu đối số
+	Phép cộng hai số	Kiểu số (Integer, Single…)	Như kiểu đối số
-	Phép trừ hai số	Kiểu số (Integer, Single…)	Như kiểu đối số
*	Phép nhân hai số	Kiểu số (Integer, Single…)	Như kiểu đối số
/	Phép chia hai số	Kiểu số (Integer, Single…)	Single hay Double
\	Phép chia lấy phần nguyên	Integer, Long	Integer, Long
Mod	Phép chia lấy phần dư	Integer, Long	Integer, Long
^	Tính lũy thừa	Kiểu số (Integer, Single…)	Như kiểu đối số

1. Các phép toán quan hệ

Đây là các phép toán mà giá trị trả về của chúng là một giá trị kiểu Boolean (TRUE hay FALSE).

Phép toán	Ý nghĩa
=	So sánh bằng nhau
<>	So sánh khác nhau
>	So sánh lớn hơn
<	So sánh nhỏ hơn
>=	So sánh lớn hơn hoặc bằng
<=	So sánh nhỏ hơn hoặc bằng

1. Các phép toán Logic: là các phép toán tác động trên kiểu Boolean và cho kết quả là kiểu Boolean. Các phép toán này bao gồm AND (và), OR (hoặc), NOT (phủ định). Sau đây là bảng giá trị của các phép toán:

X	Y	X AND Y	X OR Y	NOT X
TRUE	TRUE	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	FALSE
FALSE	TRUE	FALSE	TRUE	TRUE
FALSE	FALSE	FALSE	FALSE	TRUE

Câu lệnh

Một câu lệnh (statement) xác định một công việc mà chương trình phải thực hiện để xử lý dữ liệu đã được mô tả và khai báo. Các câu lệnh được ngăn cách với nhau bởi ký tự xuống dòng. Ký tự xuống dòng báo hiệu kết thúc một câu lệnh.

Lệnh gán

Cú pháp:

<Tên biến> = <Biểu thức>

Ví dụ:

Giả sử ta có khai báo sau:

Dim TodayTemp As Single, MinAge As Integer

Dim Sales As Single, NewSales As Single, FullName As String

Các lệnh sau gán giá trị cho các biến trên:

TodayTemp = 30.5

MinAge = 18

Sales = 200000

NewSales = Sales * 1.2

Giả sử người dùng cần nhập họ và tên vào ô nhập liệu TextBox có thuộc tính Name là txtName, câu lệnh dưới đây sẽ lưu giá trị của ô nhập liệu vào trong biến FullName:

FullName = txtName.Text

Lưu ý: Kiểu dữ liệu của biểu thức (vế phải của lệnh gán) phải phù hợp với biến ta cần gán trị.

Lệnh rẽ nhánh If

- Một dòng lệnh:

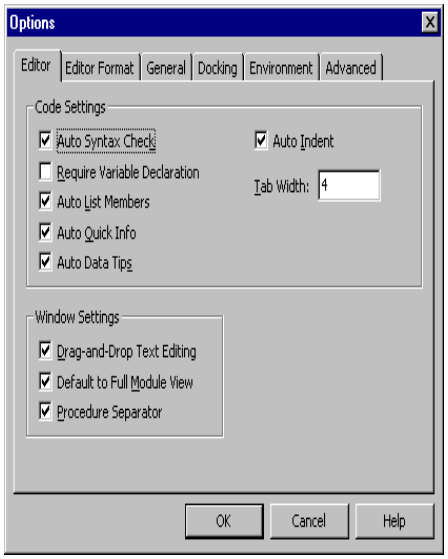
If <điều kiện> Then <dòng lệnh>

- Nhiều dòng lệnh:

If <điều kiện> Then

Các dòng lệnh

End If



Trong đó, <điều kiện>: biểu thức mà kết quả trả về kiểu Boolean.

Ý nghĩa câu lệnh: Các dòng lệnh hay dòng lệnh sẽ được thi hành nếu như điều kiện là đúng. Còn nếu như điều kiện là sai thì câu lệnh tiếp theo sau cấu trúc If ... Then được thi hành.

- Dạng đầy đủ: If ... Then ... Else

If <điều kiện 1> Then

[Khối lệnh 1]

ElseIf <điều kiện 2> Then

[Khối lệnh 2]...

[Else

[Khối lệnh n]]

End If

VB sẽ kiểm tra các điều kiện, nếu điều kiện nào đúng thì khối lệnh tương ứng sẽ được thi hành. Ngược lại nếu không có điều kiện nào đúng thì khối lệnh sau từ khóa Else sẽ được thi hành.

Ví dụ:

If (TheColorYouLike = vbRed) Then

MsgBox "You are a lucky person"

ElseIf (TheColorYouLike = vbGreen) Then

MsgBox "You are a hopeful person"

ElseIf (TheColorYouLike = vbBlue) Then

MsgBox "You are a brave person"

ElseIf (TheColorYouLike = vbMagenta) Then

MsgBox "You are a sad person"

Else

MsgBox "You are an average person"

End If

Lệnh lựa chọn Select Case

Trong trường hợp có quá nhiều các điều kiện cần phải kiểm tra, nếu ta dùng cấu trúc rẽ nhánh If…Then thì đoạn lệnh không được trong sáng, khó kiểm tra, sửa đổi khi có sai sót. Ngược lại với cấu trúc Select…Case, biểu thức điều kiện sẽ được tính toán một lần vào đầu cấu trúc, sau đó VB sẽ so sánh kết quả với từng trường hợp (Case). Nếu bằng nó thi hành khối lệnh trong trường hợp (Case) đó.

Select Case <biểu thức kiểm tra>

Case <Danh sách kết quả biểu thức 1>

[Khối lệnh 1]

Case <Danh sách kết quả biểu thức 2>

[Khối lệnh 2]

.

.

.

[Case Else

[Khối lệnh n]]

End Select

Mỗi danh sách kết quả biểu thức sẽ chứa một hoặc nhiều giá trị. Trong trường hợp có nhiều giá trị thì mỗi giá trị cách nhau bởi dấu phẩy (,). Nếu có nhiều Case cùng thỏa điều kiện thì khối lệnh của Case đầu tiên sẽ được thực hiện.

Ví dụ của lệnh rẽ nhánh If…Then ở trên có thể viết như sau:

Select Case TheColorYouLike

Case vbRed

MsgBox "You are a lucky person"

Case vbGreen

MsgBox "You are a hopeful person"

Case vbBlue

MsgBox "You are a brave person"

Case vbMagenta

MsgBox "You are a sad person"

Case Else

MsgBox "You are an average person"

End Select

Toán tử Is & To

Toán tử Is: Được dùng để so sánh <Biểu thức kiểm tra> với một biểu thức nào đó.

Toán tử To: Dùng để xác lập miền giá trị của <Biểu thức kiểm tra>.

Ví dụ:

Select Case Tuoi

Case Is <18

MsgBox “Vi thanh nien”

Case 18 To 30

MsgBox “Ban da truong thanh, lo lap than di”

Case 31 To 60

MsgBox “Ban dang o lua tuoi trung nien”

Case Else

MsgBox “Ban da lon tuoi, nghi huu duoc roi day!”

End Select

Lưu ý: Trong ví dụ trên không thể viết Case Tuoi < 18.

Cấu trúc lặp

Các cấu trúc lặp cho phép thi hành một khối lệnh nào đó nhiều lần.

1. Lặp không biết trước số lần lặp

Khối lệnhDo ... Loop: Đây là cấu trúc lặp không xác định trước số lần lặp, trong đó, số lần lặp sẽ được quyết định bởi một biểu thức điều kiện. Biểu thức điều kiện phải có kết quả là True hoặc False. Cấu trúc này có 4 kiểu:

Kiểu 1:

Do While <điều kiện>

<khối lệnh> Đkiện

Loop

Đúng Sai

Khối lệnh sẽ được thi hành đến khi nào điều kiện không còn đúng nữa. Do biểu thức điều kiện được kiểm tra trước khi thi hành khối lệnh, do đó có thể khối lệnh sẽ không được thực hiện một lần nào cả.

Kiểu 2:

Do

<khối lệnh>

Loop While <điều kiện>

Khối lệnh sẽ được thực hiện, sau đó biểu thức điều kiện được kiểm tra, nếu điều kiện còn đúng thì, khối lệnh sẽ được thực hiện tiếp tục. Do biểu thức điều kiện được kiểm tra sau, do đó khối lệnh sẽ được thực hiện ít nhất một lần.

Kiểu 3:

Do Until <điều kiện>

<khối lệnh>

Loop

Cũng tương tự như cấu trúc Do While ... Loop nhưng khác biệt ở chỗ là khối lệnh sẽ được thi hành khi điều kiện còn sai.

Kiểu 4:

Do

<khối lệnh>

Loop Until <điều kiện>

Khối lệnh được thi hành trong khi điều kiện còn sai và có ít nhất là một lần lặp.

Ví dụ: Đoạn lệnh dưới đây cho phép kiểm tra một số nguyên N có phải là số nguyên tố hay không?

Dim i As Integer

i = 2

Do While (i <= Sqr(N)) And (N Mod i = 0)

i = i + 1

Loop

If (i > Sqr(N)) And (N <> 1) Then

MsgBox Str(N) & " la so nguyen to"

Else

MsgBox Str(N) & " khong la so nguyen to"

End If

Trong đó, hàm Sqr: hàm tính căn bậc hai của một số

Lặp biết trước số lần lặp

- For ... Next

Đây là cấu trúc biết trước số lần lặp, ta dùng biến đếm tăng dần hoặc giảm dần để xác định số lần lặp.

For <biến đếm> = <điểm đầu> To <điểm cuối> [Step <bước nhảy>]

[khối lệnh]

Next

Biến đếm, điểm đầu, điểm cuối, bước nhảy là những giá trị số (Integer, Single,…). Bước nhảy có thể là âm hoặc dương. Nếu bước nhảy là số âm thì điểm đầu phải lớn hơn điểm cuối, nếu không khối lệnh sẽ không được thi hành.

Khi Step không được chỉ ra, VB sẽ dùng bước nhảy mặc định là một.

Ví dụ: Đoạn lệnh sau đây sẽ hiển thị các kiểu chữ hiện có của máy bạn.

Private Sub Form_Click( )

Dim i As Integer

For i = 0 To Screen.FontCount

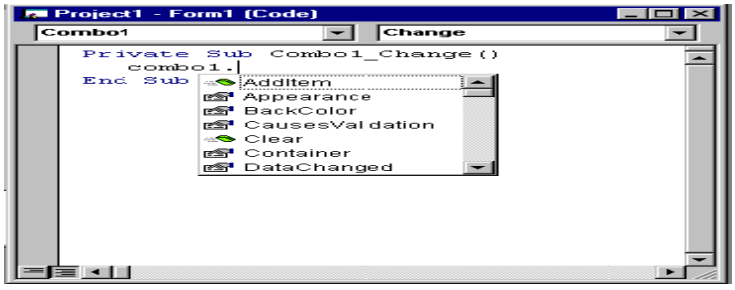
MsgBox Screen.Fonts(I)

Next

End Sub

Ví dụ: Tính N!

- TextBox: Name:txtNum Bước 1: Thiết kế chương trình có giao diện:

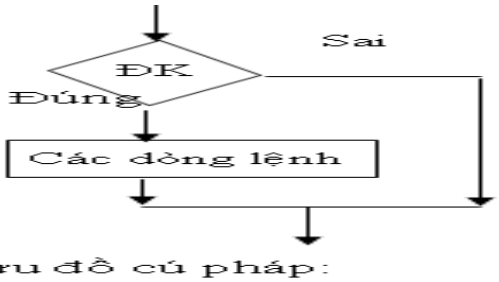


Label: Name: lblKQ

- Bước 2: Sự kiện Command1_Click được xử lý:

```
Private Sub Command1_Click()

Dim i As Integer, n As Integer, Kq As Long

n = Val(txtNum.Text)

Kq = 1

For i = 1 To n

Kq = Kq * i

Next

lblKQ.Caption = Str(Kq)

End Sub
```

- Lưu dự án và chạy chương trình ta được kết quả như hình dưới:



Lưu đồ cú pháp:

- For Each ... Next

Tương tự vòng lặp For ... Next, nhưng nó lặp khối lệnh theo số phần tử của một tập các đối tượng hay một mảng thay vì theo số lần lặp xác định. Vòng lặp này tiện lợi khi ta không biết chính xác bao nhiêu phần tử trong tập hợp.

For Each <phần tử> In <nhóm>

<khối lệnh>

Next <phần tử>

Lưu ý:

- Phần tử trong tập hợp chỉ có thể là biến Variant, biến Object, hoặc một đối tượng trong Object Browser.

- Phần tử trong mảng chỉ có thể là biến Variant.

- Không dùng For Each ... Next với mảng chứa kiểu tự định nghĩa vì Variant không chứa kiểu tự định nghĩa.

Chương trình con

Khái niệm

Trong những chương trình lớn, có thể có những đoạn chương trình viết lặp đi lặp lại nhiều lần, để tránh rườm rà và mất thời gian khi viết chương trình người ta thường phân chia chương trình thành nhiều module, mỗi module giải quyết một công việc nào đó. Các module như vậy gọi là các chương trình con.

Một tiện lợi khác của việc sử dụng chương trình con là ta có thể dễ dàng kiểm tra xác định tính đúng đắn của nó trước khi ráp nối vào chương trình chính và do đó việc xác định sai sót để tiến hành hiệu đính trong chương trình chính sẽ thuận lợi hơn.

Trong Visual Basic, chương trình con có hai dạng là hàm (Function) và thủ tục (Sub).

Hàm khác thủ tục ở chỗ hàm trả về cho lệnh gọi một giá trị thông qua tên của nó còn thủ tục thì không. Do vậy ta chỉ dùng hàm khi và chỉ khi thoả mãn đồng thời các yêu cầu sau đây:

- Ta muốn nhận lại một kết quả (chỉ một mà thôi) khi gọi chương trình con.

- Ta cần dùng tên chương trình con (có chứa kết quả) để viết trong các biểu thức.

Nếu không thỏa mãn hai điều kiện ấy thì dùng thủ tục.

Thủ tục

1. Khái niệm:

Thủ tục là một chương trình con thực hiện một hay một số tác vụ nào đó. Thủ tục có thể có hay không có tham số.

1. Khai báo thủ tục

[Private | Public] [Static] Sub <tên thủ tục> [(<tham số>[As <Kiểu tham số>])]

<Các dòng lệnh> hay <Các khai báo>

End Sub

Trong đó:

- <Tên thủ tục>: Đây là một tên được đặt giống quy tắc tên biến, hằng,…

- <tham số>[: <Kiểu tham số>]: có thể có hay không? Nếu có nhiều tham số thì mỗi tham số phân cách nhau dấu phẩy. Nếu không xác định kiểu tham số thì tham số có kiểu Variant.

Để gọi thủ tục để thực thi, ta có 2 cách:

- <Tên thủ tục> [<Các tham số thực tế>]
- Call <Tên thủ tục> ([<Các tham số thực tế>])

Ví dụ: Thiết kế chương trình kiểm tra xem số nguyên N có phải là số nguyên tố hay không?

- Bước 1: Thiết kế chương trình có giao diện

[missing_resource: .png]

TextBox: Name:txtNum

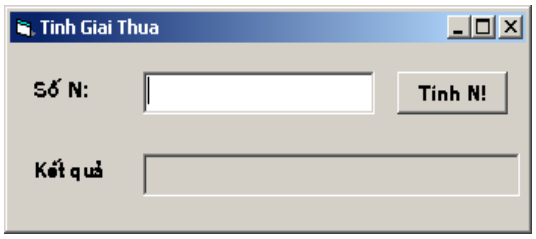
- Bước 2: Viết thủ tục KtraNgTo trong phần mã lệnh của Form

```
Sub KTraNgTo(N As Integer)

Dim i As Integer

i = 2

Do While (i <= Sqr(N)) And (N Mod i <> 0)

i = i + 1

Loop

If (i > Sqr(N)) And (N <> 1) Then

MsgBox Str(N) & " la so nguyen to"

Else

MsgBox Str(N) & " khong la so nguyen to"

End If

End Sub
```

- Bước 3: Xử lý sự kiện Command1_Click; trong thủ tục xử lý sự kiện này ta có gọi thủ tục KtraNgTo như sau:

```
Private Sub Command1_Click()
KTraNgTo Val(txtNum.Text)
‘ Call KtraNgTo(Val(txtNum.Text))
End Sub
```

- Bước 4: Lưu dự án và chạy chương trình. Ta được kết quả sau:



Trong ví dụ trên thay vì gọi thủ tục bằng lời gọi:

KTraNgTo Val(txtNum.Text)

Ta có thể sử dụng cách khác:

Call KtraNgTo(Val(txtNum.Text))

Hàm

1. Khái niệm

Hàm (Function) là một chương trình con có nhiệm vụ tính toán và cho ta một kết quả. Kết quả này được trả về trong tên hàm cho lời gọi nó.

1. Khai báo hàm

[Private | Public | Static] Function <Tên hàm> [(<tham số>[As <Kiểu tham số>])] _

[As <KIẾU DỮ LIỆU>]

<Các dòng lệnh> hay <Các khai báo>

End Function

Trong đó:

- <Tên hàm>: Đây là một tên được đặt giống quy tắc tên biến, hằng,…

- <tham số>[: <Kiểu tham số>]: có thể có hay không? Nếu có nhiều tham số thì mỗi tham số phân cách nhau dấu phẩy. Nếu không xác định kiểu tham số thì tham số có kiểu Variant.

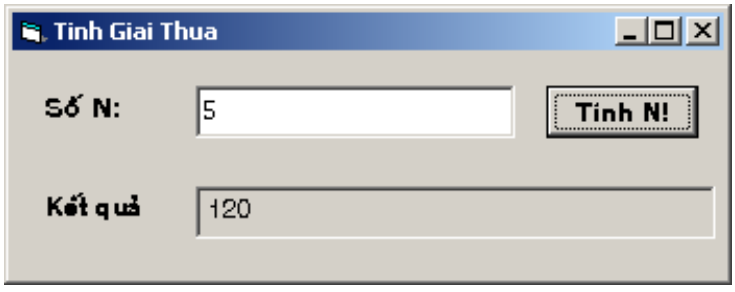
- <KIỂU DỮ LIỆU>: Kết quả trả về của hàm, trong trường hợp không khai báo As <kiểu dữ liệu>, mặc định, VB hiểu kiểu trả về kiểu Variant.

Khi gọi hàm để thực thi ta nhận được một kết quả. Cần chú ý khi gọi hàm thực thi ta nhận được một kết quả có kiểu chính là kiểu trả về của hàm (hay là kiểu Variant nếu ta không chỉ rõ kiểu trả về trong định nghĩa hàm). Do đó lời gọi hàm phải là thành phần của một biểu thức.

Cú pháp gọi hàm thực thi: <Tên hàm>[(tham số)].

Ví dụ: Tính N!

- TextBox: Name:txtNum Bước 1: Thiết kế chương trình có giao diện:



Label: Name: lblKQ

- Bước 2: Thêm một hàm vào cửa sổ mã lệnh của Form

```
Function Giaithua(N As Integer) As Long
Dim i As Integer, Kq As Long
Kq = 1
For i = 1 To n
Kq = Kq * i
Next
Giaithua = Kq
End Function
Private Sub Command1_Click()
Dim n As Integer
n = Val(txtNum.Text)
lblKQ.Caption = Str(Giaithua(n))
End Sub
```

Lưu dự án và chạy chương trình ta được kết quả như hình dưới:

Lưu ý: Do khi gọi hàm ta nhận được một kết quả nên bên trong phần định nghĩa hàm, trước khi kết thúc ta phải gán kết quả trả về của hàm thông qua tên hàm (trong ví dụ trên là dòng lệnh Giaithua = Kq)

Truy xuất dữ liệu trong Visual Basic

Các khái niệm

- Module:

- Một ứng dụng đơn giản có thể chỉ có một biểu mẫu, lúc đó tất cả mã lệnh của ứng dụng đó được đặt trong cửa sổ mã lệnh của biểu mẫu đó (gọi là Form Module). Khi ứng dụng được phát triển lớn lên, chúng ta có thể có thêm một số biểu mẫu nữa và lúc này khả năng lặp đi lặp lại nhiều lần của một đoạn mã lệnh trong nhiều biểu mẫu khác nhau là rất lớn.

- Để tránh việc lặp đi lặp lại trên, ta tạo ra một Module riêng rẽ chứa các chương trình con được dùng chung. Visual Basic cho phép 3 loại Module:

Module biểu mẫu (Form module): đi kèm với mỗi một biểu mẫu là một module của biểu mẫu đó để chứa mã lệnh của biểu mẫu này. Với mỗi điều khiển trên biểu mẫu, module biểu mẫu chứa các chương trình con và chúng sẵn sàng được thực thi để đáp ứng lại các sự kiện mà người sử dụng ứng dụng tác động trên điều khiển. Module biểu mẫu được lưu trong máy tính dưới dạng các tập tin có đuôi là *.frm.

Module chuẩn (Standard module): Mã lệnh không thuộc về bất cứ một biểu mẫu hay một điều khiển nào sẽ được đặt trong một module đặc biệt gọi là module chuẩn (được lưu với đuôi *.bas). Các chương trình con được lặp đi lặp lại để đáp ứng các sự kiện khác nhau của các điều khiển khác nhau thường được đặt trong module chuẩn.

Module lớp (Class module): được sử dụng để tạo các điều khiển được gọi thực thi trong một ứng dụng cụ thể. Một module chuẩn chỉ chứa mã

lệnh nhưng module lớp chứa cả mã lệnh và dữ liệu, chúng có thể được coi là các điều khiển do người lập trình tạo ra (được lưu với đuôi *.cls).

- Phạm vi (scope): xác định số lượng chương trình có thể truy xuất một biến. Một biến sẽ thuộc một trong 3 loại phạm vi:

Phạm vi biến cục bộ.

Phạm vi biến module.

Phạm vi biến toàn cục.

Biến toàn cục

- Khái niệm: Biến toàn cục là biến có phạm vi hoạt động trong toàn bộ ứng dụng.
- Khai báo:

Global <Tên biến> [As <Kiểu dữ liệu>]

Biến cục bộ

- Khái niệm: Biến cục bộ là biến chỉ có hiệu lực trong những chương trình mà chúng được định nghĩa.
- Khai báo:

Dim <Tên biến> [As <Kiểu dữ liệu>]

- Lưu ý:

Biến cục bộ được định nghĩa bằng từ khóa Dim sẽ kết thúc ngay khi việc thi hành thủ tục kết thúc.

Biến Module

- Khái niệm: Biến Module là biến được định nghĩa trong phần khai báo (General|Declaration) của Module và mặc nhiên phạm vi hoạt động của nó là toàn bộ Module ấy.
- Khai báo:

- Biến Module được khai báo bằng từ khóa Dim hay Private & đặt trong phần khai báo của Module.

Ví dụ:

Private Num As Integer

- Tuy nhiên, các biến Module này có thể được sử dụng bởi các chương trình con trong các Module khác. Muốn thế chúng phải được khai báo là Public trong phân Khai báo (General|Declaration) của Module.

Ví dụ:

Public Num As Integer

Lưu ý: Không thể khai báo biến với từ khóa là Public trong chương trình con.

Truyền tham số cho chương trình con

- Khái niệm

Một chương trình con đôi lúc cần thêm một vài thông tin về trạng thái của đoạn mã lệnh mà nó định nghĩa để thực thi. Những thông tin này là các biến được truyền vào khi gọi chương trình con, các biến này gọi là tham số của chương trình con.

Có hai cách để truyền tham số cho chương trình con: Truyền bằng giá trị & truyền bằng địa chỉ.

- Truyền tham số bằng giá trị

Với cách truyền tham số theo cách này, mỗi khi một tham số được truyền vào, một bản sao của biến đó được tạo ra. Nếu chương trình con có thay đổi giá trị, những thay đổi này chỉ tác động lên bản sao của biến. Trong VB, từ khóa ByVal được dùng để xác định tham số được truyền bằng giá trị.

Ví dụ:

Sub Twice (ByVal Num As Integer)

Num = Num * 2

Print Num

End Sub

Private Sub Form_Click()

Dim A As Integer

A = 4

Print A

Twice A

Print A

End Sub

Kết quả thực hiện của đoạn chương trình trên:

4

8

4

- Truyền tham số bằng địa chỉ

Truyền tham số theo địa chỉ cho phép chương trình con truy cập vào giá trị gốc của biến trong bộ nhớ. Vì thế, giá trị của biến có thể sẽ bị thay đổi bởi đoạn mã lệnh trong chương trình con. Mặc nhiên, trong VB6 các tham số được truyền theo địa chỉ; tuy nhiên ta có thể chỉ định một cách tường minh nhờ vào từ khóa ByRef.

Ví dụ:

```
Sub Twice (Num As Integer)

Num = Num * 2

Print Num

End Sub

Private Sub Form_Click()

Dim A As Integer

A = 4

Print A

Twice A

Print A

End Sub
```

Kết quả thực hiện của đoạn chương trình trên:

4

8

8

Bẫy lỗi trong Visual Basic

Các thao tác bẫy các lỗi thực thi của chương trình là cần thiết đối với các ngôn ngữ lập trình. Người lập trình khó kiểm soát hết các tình huống có thể gây ra lỗi. Chẳng hạn người ta khó có thể kiểm tra chặt chẽ việc người dùng đang chép dữ liệu từ đĩa mềm (hay CD) khi chúng không có trong ổ đĩa. Nếu có các thao tác bẫy lỗi ở đây thì tiện cho người lập trình rất nhiều.

Visual Basic cũng cung cấp cho ta một số cấu trúc để bẫy các lỗi đang thực thi.

Cú pháp:

Dạng 1:

On Error GoTo <Tên nhãn>

<Các câu lệnh có thể gây ra lỗi>

<Tên nhãn>:

<Các câu lệnh xử lý lỗi>

Ý nghĩa:

- <Tên nhãn>: là một tên được đặt theo quy tắc của một danh biểu.

- Nếu một lệnh trong <Các câu lệnh có thể gây ra lỗi> thì khi chương trình thực thi đến câu lệnh đó, chương trình sẽ tự động nhảy đến đoạn chương trình định nghĩa bên dưới <Tên nhãn> để thực thi.

Dạng 2:

On Error Resume Next

<Các câu lệnh có thể gây ra lỗi>

Ý nghĩa:

- Nếu một lệnh trong <Các câu lệnh có thể gây ra lỗi> thì khi chương trình thực thi đến câu lệnh đó, chương trình sẽ tự động bỏ qua câu lệnh bị lỗi và thực thi câu lệnh kế tiếp.

Các kiểu dữ liệu có cấu trúc
Chương này giới thiệu về các cấu trúc dữ liệu trong VB. Việc nắm bắt được các vấn đề này giúp cho việc tổ chức dữ liệu khi viết chương trình VB được hợp lý hơn.

Học xong chương này, sinh viên phải nắm bắt được các vấn đề sau:

- Sử dụng kiểu dữ liệu chuỗi.

- Sử dụng kiểu ngày tháng.

- Kiểu động (Variant)

- Kiểu mảng

Kiến thức có liên quan:

- Các cấu trúc lập trình trong VB.

Tài liệu tham khảo:

http://www.vovisoft.com/VisualBasic/VB6Chapter5.htm

http://www.vovisoft.com/VisualBasic/VB6Chapter6.htm

Kiểu chuỗi ký tự (String)

Khai báo

Có hai đặc tả chuỗi ký tự theo cú pháp như sau:

- String * <Chiều dài> Chỉ ra một chuỗi ký tự có độ dài cố định là bao nhiêu ký tự. Trong trường hợp giá trị thực của chuỗi có độ dài ngắn hơn độ dài khai báo thì độ dài của chuỗi thì một số khoảng trắng được thêm vào cho đủ độ dài thực. Trong trường hợp giá trị thực của chuỗi có độ dài lớn hơn độ dài khai báo thì sẽ cắt bớt các ký tự dư thừa bên phải. Một chuỗi không có ký tự nào (độ dài bằng 0) gọi là chuỗi rỗng.

- String: Khi không chỉ ra chiều dài tối đa của chuỗi thì mặc nhiên chuỗi có chiều dài tối đa là 65.500 ký tự.

Ví dụ:

Dim Name As String * 30, Class As String * 10

Dim A As String

Các hàm xử lý chuỗi

- Ghép chuỗi: cho phép ghép 2 hay nhiều chuỗi lại với nhau nhờ phép toán &.

Ví dụ:

Dim FirstWord As String, SecondWord As String

Dim Greeting As String

FirstWord = "Hello"

SecondWord = "World"

Greeting = FirstWord & SecondWord

' Greeting bây giờ là "HelloWorld"

- Len: trả về chiều dài một chuỗi được chỉ định.

Ví dụ:

Greeting = "Hi John!"

Dim iLen As Integer

iLen = Len(Greeting) ' iLen bây giờ bằng 8

- Left: Trích chuỗi con từ phần đầu chuỗi gốc Left (String, [length]).
- Right: Trích chuỗi con từ phần đuôi chuỗi gốc Right (String, [length])
- Mid: Trích chuỗi con từ giữa chuỗi gốc

Mid(String, Start As Long, [length])

Ví dụ 1:

Dim Today As String, StrDay As String, StrMonth As String

Dim StrYear As String, StrMonthYear As String

Today = "24/05/2001" ' Lấy ra 2 ký tự từ bên trái của chuỗi TodayStrDay = Left(Today,2) ' StrDay bây giờ bằng "24"' Lấy ra 4 ký tự từ bên phải của String TodayStrYear = Right(Today,4) ' StrYear bây giờ bằng "2001"' Lấy ra 2 characters bắt đầu từ ký tự thứ tư của chuỗi

‘ Today, ký tự đầu tiên từ bên trái là thứ nhấtStrMonth = Mid(Today,4,2) ' StrMonth bây giờ bằng "05"' Lấy ra phần còn lại bắt đầu từ ký tự 4 của chuỗi TodayStrMonthYear = Mid(Today,4) ' StrMonthYear bằng “05/2001"

Ví dụ 2:

Today = "24/05/2001"

' Thay thế character thứ 3 của Today bằng "-"

Mid(Today,3,1) = "-"

' Thay thế 2 ký tự bắt đầu từ ký tự 4 của Today bằng "10"

Mid(Today,4,2) = "10"

' Thay thế character thứ 6 của Today bằng "-"

Mid(Today,6,1) = "-" ' Today bây giờ bằng "24-10-2001"

- InStr: Tìm chuỗi con trong chuỗi gốc. Nếu hàm InStr trả về 0, nghĩa là không tìm thấy.

Cú pháp: InStr([start,] string1, string2 [, compare])

Trong đó:

- Start: Xác định vị trí trong chuỗi bắt đầu việc tìm kiếm. Nếu giá trị là Null thì sẽ bắt đầu từ đầu chuỗi. Nếu như tham số Compare có đặc tả thì bắt buộc phải khai báo tham số Start.

- String1: Biểu thức chuỗi để so sánh.

- String2: Chuỗi cần tìm.

- Compare: Xác định kiểu so sánh chuỗi.

Giá trị: vbTextCompare, vbBinaryCompare.

Ví dụ 1:

```
Dim myString As String, Position As Integer

myString = "The *rain in Spain mainly..."

Position = Instr(myString,"*") ' Position sẽ là 5

Nếu trong myString không có dấu "*" thì Position sẽ bằng 0
```

Ví dụ 2:

```
Dim KeyValuePair As String, Key As String

Dim Value As String

KeyValuePair = "BeatlesSong=Yesterday"

Pos = Instr(KeyValuePair, "=")
```

Key = Left(KeyValuePair, Pos-1)

Value = Mid(KeyValuePair, Pos+1)

- Replace: tìm và thay thế chuỗi.

Cú pháp:

Replace(Expression, find, replace[, start[, count[, compare]]])

Trong đó:

- Expression:Biểu thức chuỗi chứa chuỗi cần thay thế.

- find:Chuỗi cần tìm.

- replace: Chuỗi thay thế chuỗi tìm được.

- start:Tương tự như hàm InStr.

- count: Xác định số lần thay thế. Mặc định là 1.

- compare: Tương tự như hàm InStr.

- LTrim (RTrim): cắt tất cả các khoảng trắng bên trái (bên phải của chuỗi)

Cú pháp:LTrim(string)

RTrim(string)

- UCase: đổi chuỗi sang chuỗi gồm các ký tự là chữ hoa.

Cú pháp:UCase(string)

- Asc: cho mã Ascii của một ký tự.
- Chr: trả về ký tự ứng với mã Ascii được chỉ định.

Dim ASCIINumberA As Integer, CharB As String * 1

Dim StrFive As String * 1

ASCIINumberA = Asc("A") ' ASCIINumberA bây giờ bằng 65

CharB = Chr(66)

StrFive = Chr(Asc("0") + 5) ' ta có digit "5"

- InstrRev: tương tự như InStr nhưng việc tìm kiếm được tiến hành từ phải sang.
- Val: Hàm đổi chuỗi sang số.
- Str: Hàm đổi số sang chuỗi.

Kiểu ngày tháng (Date)

- Là kiểu mà các biến của nó chứa giá trị ngày tháng.

- Để cho VB biết dữ liệu là kiểu Date ta cần đặt giữa hai dấu # (hoặc cặp “”).

Ví dụ:

Dim D As Date

D = #01/02/98# ‘ Hay “01/02/98”

[missing_resource: .png]

Hình IV.1 Hộp thoại xác lập kiểu ngàyNếu hiểu theo kiểu người Mỹ, đây là ngày 2 tháng giêng năm 1998, còn nếu theo kiểu Anh thì đây là ngày 1 tháng hai năm 1998. Tuy nhiên, định dạng ngày tháng hiển thị phụ thuộc vào quy định của Windows.

- Hộp thoại hình IV.1 hiển thị khi ta chọn Regional Setting trong cửa sổ Control Panel của Windows, nó cho phép quy định kiểu ngày tháng tùy

thuộc cách mà người dùng quy định. VB xử lý ngày tháng theo kiểu Mỹ, nhưng nếu máy hiển thị theo kiểu Anh thì nó vẫn hiển thị theo kiểu Anh.

- Hàm Now: trả về ngày giờ hiện tại.

Ví dụ: Dùng hàm Now & Format:

MsgBox "NOW IS " & Format (Now, "ddd dd-mmm-yyyy hh:nn:ss")

' sẽ hiển thị

NOW IS Tue 05-Oct-2004 16:15:53

Các loại số

- Để chuyển đổi một chuỗi ra số ta có các hàm Val, CInt, CSng. Ngược lại để chuyển đổi từ số sang chuỗi ta dùng CStr, Str.

Ví dụ:

Dollars = "500"

ExchangeRatePerDollar = "7000"

tempValue= Val(Dollars) * Val(ExchangeRatePerDollar)

VNDong = CStr(tempValue)

MsgBox "Amount in VN Dong is " & VNDong

Ví dụ:

Dollars = "500.0" ExchangeRatePerDollar = "7000.0"'Dùng hàm CSng để đổi chuỗi ra SingletempValue = CSng(Dollars) * CSng(ExchangeRatePerDollar)'Dùng hàm Format để có các dấu phẩy ở ngàn và triệu

‘ và phải có 2 chữ số sau dấu chấm thập phân.VNDong = Format (tempValue, "#,###,###.00")MsgBox "Amount in VN Dong is " & VNDong

- Round: bỏ bớt một số chữ số sau dấu chấm thập phân

Ví dụ:

Round (12.3456789, 4)

chỉ giữ lại 4 con số sau dấu chấm thập phân và cho ta 12.3457

Kiểu Object

Biến kiểu Object chứa một địa chỉ 4 Byte trỏ đến đối tượng trong ứng dụng hiện hành hoặc các ứng dụng khác. Dùng lệnh Set để chỉ ra đối tượng cụ thể.

Dim ObjDb As Object

Set ObjDb = OpenDatabase("d:\tqdinh\thu.mdb")

Khi khai báo biến đối tượng, ta nên chỉ ra tên lớp tường minh, chẳng hạn như TextBox thay vì Control, ứng dụng của ta sẽ chạy nhanh hơn.

Ta có thể xem danh sách các lớp có sẵn trong cửa sổ Object Browser.

Kiểu Variant

Biến kiểu Variant có thể chứa mọi kiểu dữ liệu kể cả kiểu mảng, kiểu do người dùng định nghĩa nhưng ngoại trừ kiểu chuỗi có độ dài cố định .

Biến kiểu Variant có thể nhận các giá trị đặc biệt như Empty, Nothing, Error, Null. Ta có thể xác định kiểu dữ liệu của biến Variant bằng các sử dụng hàm VarType hoặc hàm TypeName.

Hàm VarType dùng để kiểm tra kiểu dữ liệu

Hằng	Giá trị	Diễn giải
vbEmpty	0	Không chứa gì cả
vbNull	1	Dữ liệu không hợp lệ
vbInteger	2	Dữ liệu kiểu Integer chuẩn
vbLong	3	Dữ liệu kiểu Long Integer
vbSingle	4	Dữ liệu kiểu dấu chấm động Single
vbDouble	5	Dữ liệu kiểu dấu chấm động Double
vbCurrency	6	Kiểu Currency
vbDate	7	Kiểu Date
vbString	8	Kiểu String
vbObject	9	Kiểu Object
vbError	10	Có một đối tượng lỗi
vbBoolean	11	Kiểu giá trị Boolean chuẩn
vbVariant	12	Kiểu Variant
vbDataObject	13	Kiểu DAO chuẩn (data access object)
vbDecimal	14	Giá trị thuộc hệ thập phân

vbByte	17	Kiểu Byte
vbUserDefinedType	36	Kiểu do người dùng định nghĩa
vbArray	8192	Kiểu mảng

Một số chú ý khi dùng biến kiểu Variant:

- Nếu muốn thi hành các hàm toán học, Variant phải chứa giá trị kiểu số.

- Nếu muốn nối chuỗi, dùng toán tử & thay vì toán tử +.

Giá trị Empty:

- Đây là giá trị đặc biệt xuất hiện khi một biến chưa được gán trị. Ta dùng hàm IsEmpty để kiểm tra giá trị Empty.

- Giá trị Empty biến mất khi có một giá trị bất kỳ được gán cho biến Variant, để trở về giá trị Empty, ta gán từ khoá Empty cho biến Variant.

Giá trị Null: Biến Variant chứa giá trị Null trong trường hợp những ứng dụng cơ sở dữ liệu thể hiện không có dữ liệu hoặc dữ liệu không xác định.

Giá trị Error: Trong một biến kiểu Variant, Error là một giá trị đặc biệt cho biết đã có một lỗi đã xảy ra bên trong thủ tục.

Ví dụ:

```
Private Sub cmdShowDataTypes_Click()

Dim sMess As String

Dim vVariant As Variant

vVariant = "Xin chao" 'String

sMess = VarType(vVariant) & vbCrLf ' xuống dòng & về đầu dòng
```

```
vVariant = 25 ' Integer

sMess = sMess & VarType(vVariant) & vbCrLf

vVariant = True ' Boolean

sMess = sMess & VarType(vVariant) & vbCrLf

'Date

vVariant = #1/1/2001# 'trong cặp dấu #

sMess = sMess & VarType(vVariant)

MsgBox sMess

End Sub
```

Khi chạy chương trình kết quả là:

[missing_resource: graphics1.jpg]

Kiểu Mảng

Khái niệm

- Mảng là tập hợp các phần tử có cùng một kiểu.

- Dùng mảng sẽ làm cho chương trình đơn giản và gọn hơn vì ta có thể sử dụng vòng lặp. Mảng sẽ có biên trên và biên dưới, trong đó các thành phần của mảng là liên tiếp trong khoảng giữa hai biên này.

- Có hai loại biến mảng: mảng có chiều dài cố định và mảng có chiều dài thay đổi lúc thi hành.

Khai báo

- Mảng có chiều dài cố định:

Dim <Tên biến mảng>(<Kích thước>) [As <Kiểu>]

Lúc này phần tử đầu tiên có chỉ số là 0 & phần tử cuối cùng có chỉ số là <Kích thước>.

Dim <Tên biến mảng>(<Chỉ số đầu> To <Chỉ số cuối>) [As <Kiểu>]

Ví dụ:

' Khai báo một biến mảng 15 phần tử kiểu Integer

Dim Counters(14)As Integer

' Khai báo một biến mảng 21 phần tử kiểu Double

Public Sums(20)As Double

' Khai báo một biến mảng 10 phần tử kiểu chuỗi ký tự

Dim List (1 To 10) As String * 12

- Hàm UBound trả về biên trên của một mảng.

- Hàm LBound trả về biên dưới của một mảng.

Ví dụ:

UBound(List) sẽ trả về giá trị là 10.

LBound(List) sẽ trả về giá trị là 1.

- Lưu ý: ta có thể khai báo một mảng nhiều chiều như sau

Dim Multi3D (3, 1 To 10, 9) As Double

Khai báo này tạo ra một mảng 3 chiều với kích thước 4 x 10 x 10.

- Mảng động:

- Đây là mảng có kích thước thay đổi, đó là một trong những ưu điểm của mảng động vì nó giúp ta tiết kiệm tài nguyên hệ thống. Ta có thể sử dụng một mảng có kích thước lớn trong một thời gian nào đó rồi xoá bỏ để trả lại vùng nhớ cho hệ thống.

- Khai báo một mảng động bằng cách cho nó một danh sách không theo chiều nào cả. Cú pháp: Dim <Tên mảng> () [As <Kiểu>]

Ví dụ:

Dim DynArray() As Integer

Sau đó ta có thể cấp phát số phần tử thật sự bằng lệnh ReDim.

ReDim <Tên mảng>(N) ' Trong đó N là một biểu thức kiểu Integer.

ReDim dùng để xác định hay thay đổi kích thước của một mảng động. Ta có thể dùng ReDim để thay đổi số phần tử, số chiều của một mảng nhiều lần nhưng không thể thay đổi kiểu dữ liệu của mảng ngoại trừ kiểu mảng là kiểu Variant.

Mỗi lần gọi ReDim tất cả các giá trị chứa trong mảng sẽ bị mất. VB khởi tạo lại giá trị cho chúng (Empty đối với mảng Variant, 0 cho mảng kiểu số, chuỗi rỗng cho mảng chuỗi hoặc Nothing cho mảng các đối tượng). Nhưng đôi khi ta muốn tăng kích cỡ của mảng nhưng không muốn làm mất dữ liệu, ta dùng ReDim đi kèm với từ khoá Preserve. Ta xem ví dụ dưới đây:

ReDim Preserve DynArray (UBound(DynArray) +10)

Tuy nhiên chỉ có biên trên của chiều cuối cùng trong mảng được thay đổi khi ta dùng Preserve. Nếu ta cố tình thay đổi chiều khác hoặc biên dưới thì VB sẽ báo lỗi.

Một số thao tác trên mảng

- Truy xuất từng phần tử trong mảng: <Tên mảng>(<Vị trí>)
- Sao chép mảng: Đối với VB6, ta có thể gán một mảng cho một mảng khác, hoặc kết quả trả về của một hàm có thể là một mảng.

Ví dụ:

Sub ByteCopy (old () As Byte, New () As Byte)

New = old

End Sub

Tuy nhiên, cách này cũng chỉ áp dụng được cho mảng khai báo động mà thôi. Khi gán biến, có một số quy luật mà ta cần lưu ý: Đó là quy luật về kiểu dữ liệu và quy luật về kích thước và số chiều của mảng.

Lỗi khi gán mảng có thể xảy ra lúc biên dịch hoặc khi thi hành. Ta có thể thêm bẫy lỗi để đảm bảo rằng hai mảng là tương thích trước khi gán.

- Mảng là kết quả trả về của hàm. Chẳng hạn như:

Public Function ArrayFunction (b As Byte) As Byte()

Dim x(2) As Byte

x(0) = b

x(1) = b + 2

x(2) = b + b

ArrayFunction = x

End Function

Khi gọi hàm trả về mảng, biến giữ giá trị trả về phải là một mảng và có kiểu như kiểu của hàm, nếu không nó sẽ báo lỗi "không tương thích

kiểu".

Kiểu do người dùng định nghĩa - Kiểu mẩu tin

Cú pháp:

Type <tên kiểu>

<Tên trường 1> : <Kiểu trường 1>

<Tên trường 2> : <Kiểu trường 2>

:

<Tên trường n> : <Kiểu trường n>

End Type

Ví dụ:

Type TEmployee

Fullname As String

Salary As Single

Age As Integer

End Type

Chúng ta vừa định nghĩa một kiểu dữ liệu mới có tên là TEmployee. Kiểu này có nét tương tự như một lớp. Về mặt chức năng, cả hai là như nhau, nhưng một lớp có thể chứa trong DLL và sẵn sàng cho việc dùng chung với các ứng dụng khác, trong khi đó kiểu dữ liệu do người dùng định nghĩa phải được khai báo lại trong từng dự án. Do vậy, kiểu lớp có nhiều mặt tiện lợi hơn.

Cách truy xuất từng trường của kiểu mẩu tin:

<Tên biến mẩu tin>.<Tên trường>

Ví dụ: Giả sử ta có khai báo biến sau:

Dim e As TEmployee

Ta có thể gán:

e.Fullname = “Nguyen Van An”

e.Salary = 300000.00

e.Age = 26

Câu lệnh With:

- Được sử dụng để viết gọn hơn khi thao tác với dữ liệu kiểu mẩu tin.

- Cú pháp:

With <Tên biến mẩu tin>

[Truy xuất đến từng trường của mẩu tin theo dạng:

.<Tên trường>

]

End With

Ví dụ: Dim e As TEmployee

Ta có thể gán:

With e

.Fullname = “Nguyen Van An”

.Salary = 300000.00

.Age = 26

End With

Thiết kế biểu mẫu dùng các điều khiển
Mục tiêu: Chương này gồm các bài tập nhằm rèn luyện cho sinh viên các thao tác cần thiết cho phép thiết kế các ứng dụng đơn giản trong môi trường lập trình Visual Basic cũng như một số kỹ năng lập trình cơ bản khi làm việc với Visual Basic.

Học xong chương này, sinh viên phải nắm bắt được các vấn đề sau:

- Sử dụng các điều khiển để thiết kế giao diện trong Visual Basic.

- Vận dụng các cấu trúc lập trình trong Visual Basic để viết mã lệnh.

- Sử dụng một số cấu trúc dữ liệu trong Visual Basic.

Kiến thức có liên quan:

- Giáo trình “Visual Basic”; Chương 1, 2, 3, 4, 5.

Tài liệu tham khảo:

- Visual Basic 6 Certification Exam Guide - Chapter 1, Page 1; Chapter 2, Page 41; Chapter 4, Page 89 - Dan Mezick & Scot Hillier - McGraw-Hill - 1998.

SỬ DỤNG MỘT SỐ ĐIỀU KHIỂN

Bài tập có hướng dẫn

Bài tập 1I-1

THAO TÁC TRÊN LISTBOX

Bước 1: Tạo thư mục Basic\Bt1-1. Tạo một dự án mới kiểu Standard EXE, lưu vào trong thư mục trên.

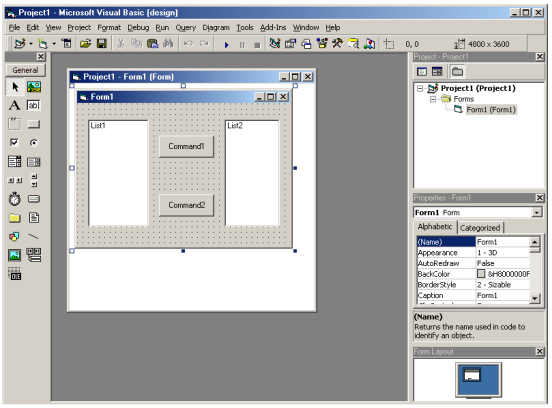
Bước 2: Thêm 2 List Box và một Button vào form (hình 1). Nhấn đúp lên form để mở ra cửa sổ Code, nhập các đoạn mã sau trong sự kiện

Form_Load:

Form1.List1.AddItem “Thing 3”

Form1.List1.AddItem “Thing 2”

Form1.List1.AddItem “Thing 1”



Hình 1.1: Thao tác với List Box

Bước 3: Chạy ứng dụng bằng cách chọn Run/Start. List1 hiển thị 3 phần tử vừa thêm vào ở bước 2. Chấm dứt chương trình bằng cách chọn Run/End trên menu để trở về môi trường soạn thảo.

Bước 4: Nhấp đúp lên Button Command1 để hiển thị sự kiện Click của Command1.

Bước 5: Mục đích của Command1 là chuyển những phần tử được chọn từ List1 sang List2. Thêm đoạn mã sau vào thủ tục sự kiện Click của Command1:

```
' Kiem tra neu mot phan tu duoc chon

If Form1.List1.ListIndex = -1 Then Exit Sub

' Chep cac phan tu duoc chon tu List1 sang List2
```

Form1.List2.AddItem Form1.List1.List(Form1.List1.ListIndex)

Bước 6: Chạy ứng dụng. Nhấp phần tử thứ nhất của List1, sau đó nhấp Command1. Điều gì xảy ra? Phần tử được chọn của List1 phải được hiển thị bên List2. Chấm dứt ứng dụng và trở về môi trường soạn thảo.

Bước 7: Tìm trong phần trợ giúp các thuộc tính sau của ListBox:

oListCount

oList

oListIndex

Bước 8: Tìm trong phần trợ giúp các hàm sau của ListBox:

oAddItem

oRemoveItem

oClear

Bước 9: Tìm trợ giúp cho lệnh VB:

Exit Sub

Bước 10: Đoạn mã trong thủ tục Command1_Click thực hiện thao tác chép phần tử từ một ListBox sang một ListBox khác. Bây giờ ta làm ngược lại: loại bỏ phần tử trong List1. Để làm điều này ta nhấp đúp lên Command1 và thêm dòng code sau vào cuối thủ tục:

' Xoa phan tu duoc chon trong List1

Form1.List1.RemoveItem Form1.List1.ListIndex

Bước 11: Chạy chương trình và chọn phần tử thứ nhất trong List1. Điều gì xảy ra?

Bước 12: Nếu không chọn phần tử nào trong List1, nhấp Command1. Điều gì xảy ra? Tại sao?

Bước 13: Ta đã có một button dùng để chuyển các phần tử được lựa chọn từ trái sang phải (List1 sang List2), với button còn lại ta sẽ dùng để chuyển các phần tử được chọn từ phải sang trái (List2 sang List1).

Bước 14: Với Command2 ta sẽ copy đoạn mã từ Command1 với 1 vài thay đổi nhỏ.

Bước 15: Command2 thực hiện các thao tác giống với Command1, nhưng có nhiệm vụ di chuyển phần tử được lựa chọn từ List2 sang List1. Đoạn mã trong Command1 sẽ được sử dụng lại với một vài thay đổi nhỏ. Nhấp đúp lên Command1, chọn các mã lệnh đã thêm vào ở các bước trước. Chọn Edit/Copy trên menu.

Bước 16: Đóng cửa sổ Code và nhấp đúp lên Command2. Sự kiện Command2_Click sẽ hiển thị trong cửa sổ Code. Nhấp bất kỳ bên trong thủ tục sự kiện và chọn Edit/Paste trên menu. Như vậy ta đã chép đoạn mã từ Command1 sang Command2.

Bước 17: Sửa lại các mã lệnh vừa được chép. Thay đổi các chú thích cho thích hợp; đổi List1 thành List2 và ngược lại. Những sửa đổi này giúp Command2 có thể thực hiện thao tác chuyển các phần tử được chọn từ List2 sang List1.

Lưu các công việc đã thực hiện bằng cách chọn File/Save Project.

Bước 18: Chạy chương trình. Chọn phần tử thứ nhất trong List1 và chọn Command1 để chuyển nó sang List2. Bây giờ chọn phần tử thứ nhất trong List2, và nhấp Command2. Nếu Command2 không thực thi, trở lại môi trường soạn thảo. Kiểm tra lại đoạn mã lệnh trong thủ tục Command2_Click ta vừa chép ở bước trên.

Bước 19: Lưu ý rằng các phần tử ở cả 2 ListBox không được sắp thứ tự; nếu muốn sắp thứ tự, ta nhấp List1 và đổi thuộc tính Sorted thành True, tương tự đối với List2.

Bước 20: Lưu dự án lại và chạy chương trình. Tất cả các phần tử phải được hiển thị theo thứ tự trong cả 2 ListBox, bất chấp thứ tự chúng được thêm vào trong ListBox.

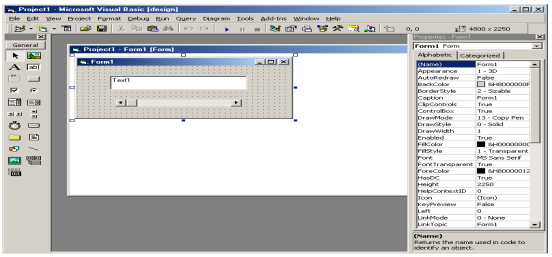
Bài tập 1I-2

THAO TÁC VỚI SPINCONTROL

Một SpinControl là sự kết hợp của TextBox và Slider. Slider tạo một miền giá trị số được hiển thị trong TextBox. Các giá trị này có thể được thay đổi bằng cách nhập trực tiếp vào trong TextBox.

Bước 1: Tạo thư mục Basic\Bt1-2. Tạo dự án mới trong thư mục trên.

Bước 2: Trong Form1, thêm một TextBox và Horizontal Scroll Bar như hình 2. Thiết lập các thuộc tính sau cho mỗi Control:



Hình I.2: Spin Control

Item1: TextBox

Name: Text1

Text: <blank>

Item2: Horizontal Scroll Bar

Name: Hscroll1

LargeChange: 10

Max: 100

Bước 3: Nhấp đúp lên scrollbar để nhập mã lệnh, đây là sự kiện Change của Scroll Bar gọi là hàm HScroll1_Change. Thêm đoạn mã sau để hiển thị giá trị hiện thời của scroll bar trong TextBox.

Text1.Text = HScroll1.Value

Bước 4: Chạy ứng dụng bằng cách chọn Run/Start trên menu. Bây giờ nhấp các mũi tên trái và phải của scroll bar. Giá trị trong TextBox phải thay đổi.

Bước 5: Bây giờ thêm mã để thay đổi giá trị bằng cách nhập trực tiếp giá trị trong TextBox. Nhấp đúp vào TextBox và thêm đoạn mã sau để thiết lập giá trị cho scroll bar khi TextBox thay đổi:

HScroll1.Value = Text1.Text

Bước 6: Chạy chương trình và nhập 50 vào TextBox. Vạch của scroll bar thay đổi theo. Thay đổi vạch của scroll bar, giá trị trong TextBox cũng thay đổi.

Bước 7: Trong khi chạy chương trình, nhập ký tự A vào TextBox. Điều gì xảy ra? Nguyên nhân vì scroll bar chỉ nhận các giá trị là số chứ không phải ký tự.

Bước 8: Để ngăn chặn những ký tự không mong muốn được nhập vào TextBox, ta sử dụng sự kiện KeyPress. Sự kiện này xảy ra khi có một phím trên bàn phím được nhấn, nhưng trước khi giá trị thực sự được hiển thị trên TextBox. Sự kiện này nhận một giá trị số nguyên của phím được nhấn, gọi là ASCII. Mỗi ký tự trên bàn phím được đại diện bằng một mã ASCII duy nhất. Do đó ta có thể kiểm tra phím nào được nhấn và bỏ qua nó nếu ta thấy không cần thiết.

Bước 9: Thêm đoạn mã sau vào sự kiện Text1_KeyPress để ngăn chặn các giá trị không phải là số.

```
' Loai bo ky tu khong can thiet

If KeyAscii = vbKeyBack Then Exit Sub

If KeyAscii < vbKey0 Or KeyAscii > vbKey9 Then

KeyAscii = 0

End If
```

Bước 10: Lưu dự án lại và chạy chương trình.

Bài tập 1I-3

THAO TÁC VỚI DRIVELISTBOX, DIRLISTBOX, FILELISTBOX

Trong ví dụ này ta phải tạo 5 đối tượng, trong đó có 4 điều khiển:

- Một Form.
- Một điều khiển DriveListBox
- Một điều khiển DirListBox
- Một điều khiển FileListBox
- Một điều khiển ImageBox

Bước 1: Tạo giao diện người dùng. Ta chỉ cần nhấp và vẽ đúng vị trí từng điều khiển trên Form.

[missing_resource: .png]

1243

Hình I.3: Giao diện lựa chọn tập tin hình ảnh để hiển thị

1: DriveListBox

Name: drvSource

2:DirListBox

Name: dirSource

3:FileListBox

Name: filSource

Pattern: *.bmp;*.wmf;*.ico;*.jpg

4:ImageBox

Name: ImgSource

Stretch: TRUE

Bước 2: Viết mã trao đổi thông tin giữa các đối tượng:

Trong cửa sổ thiết kế Form, nhấp đúp vào DriveListBox, cửa sổ Code hiện ra, xử lý sự kiện sau:

```
Private Sub drvSource_Change()

dirSource.Path = drvSource.Drive

End Sub
```

Tương tự cho DirListBox & FileListBox

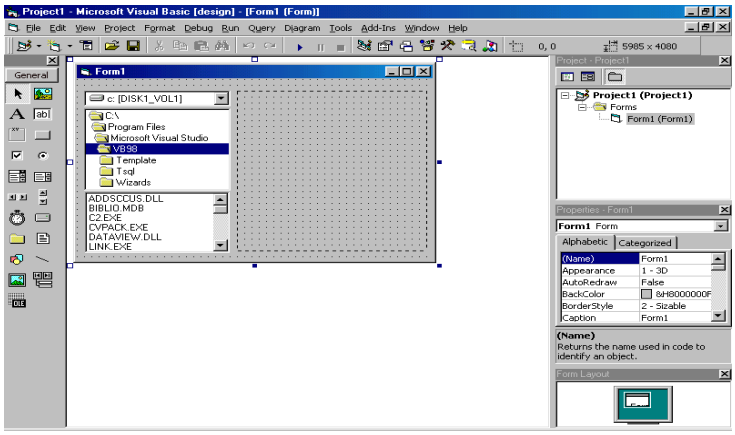
```
Private Sub dirSource_Change()

filSource.Path = dirSource.Path

End Sub
```

Private Sub filSource_Click()

imgSource.Picture = LoadPicture(filSource.Path & "\" & filSource.FileName)

End Sub

Bước 3: Lưu dự án lại vào thư mục Basic\Bt1-3. Chạy chương trình nhờ phím F5.



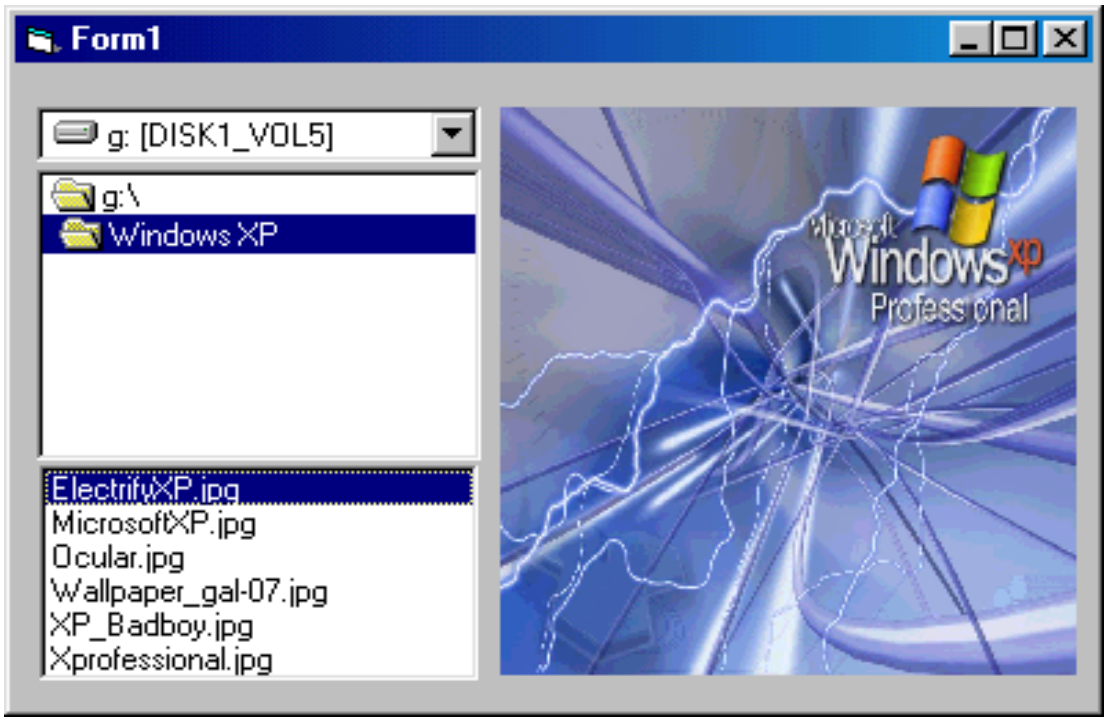
Hình I.4: Kết quả thực thi

Bài tập 1I-4

ĐIỀU KHIỂN OLE

Bước 1: Tạo dự án mới, trong đó ta có sử dụng OLE.

Hộp thoại Insert Object hiện ra để ta lựa chọn, ở đây chọn kích hoạt Create New, Object Type là Bitmap Image; đánh dấu chọn Display as Icon.



Hình I.5: Sử dụng OLE Control

Bước 2: Nhấp OK, VB sẽ gọi trình ứng dụng Paint & ta vẽ hình trên cửa sổ Paint. Sau đó chọn Exit & Return trong cửa sổ Form, ta được:

[missing_resource: .png]

[missing_resource: .png]

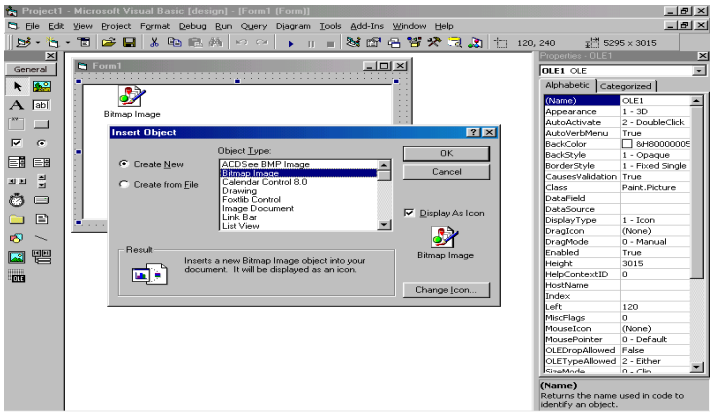
Hình I.6: Kết quả thực thi ứng dụng

Bước 3: Lưu dự án vào thư mục Basic\Bt1I-4 và chạy chương trình; nhấp đúp vào biểu tượng Bitmap Image,VB sẽ khởi động Paint để ta hiệu chỉnh hình vẽ đầu.

Bài tập tự làm

1. Thiết kế chương trình như sau:

Hình I.7 Các phép tính cơ bản



Nhập vào 2 giá trị A, B; sau đó chọn một phép toán (+,-,*,/). Nhấp chọn nút nhấn Thực hiện, kết quả sẽ hiển thị trong điều khiển nhãn Kết quả.

1. Thiết kế chương trình để nhập vào tọa độ của hai điểm (x1,y1); (x2,y2) và cho phép:

a) Tính hệ số góc của đường thẳng đi qua hai điểm đó theo công thức:

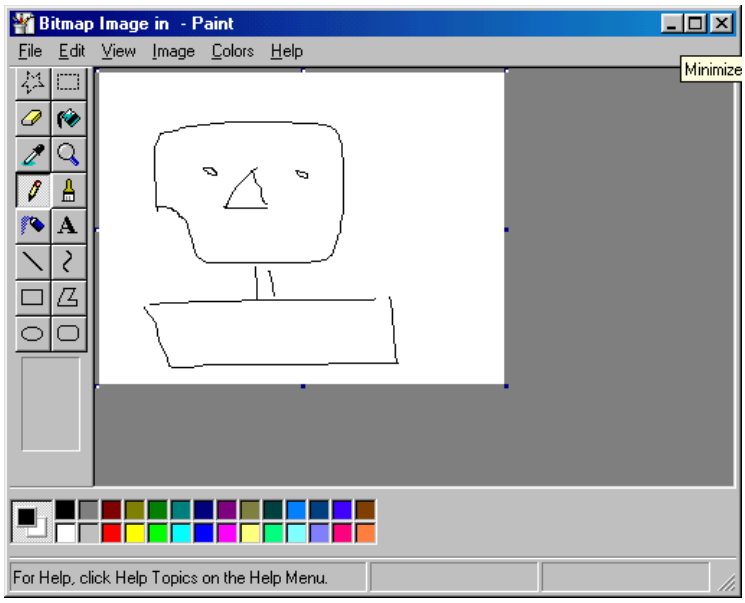
Hệ số góc = (y2 - y1) /(x2 - x1)

b) Tính khoảng cách giữa hai điểm theo công thức:

khoảng cách = $\sqrt{(x2 - x1)^2 + (y2 - y1)^2}$

Giao diện chương trình có thể như sau:

Hình I.8: Tọa độ các điểm



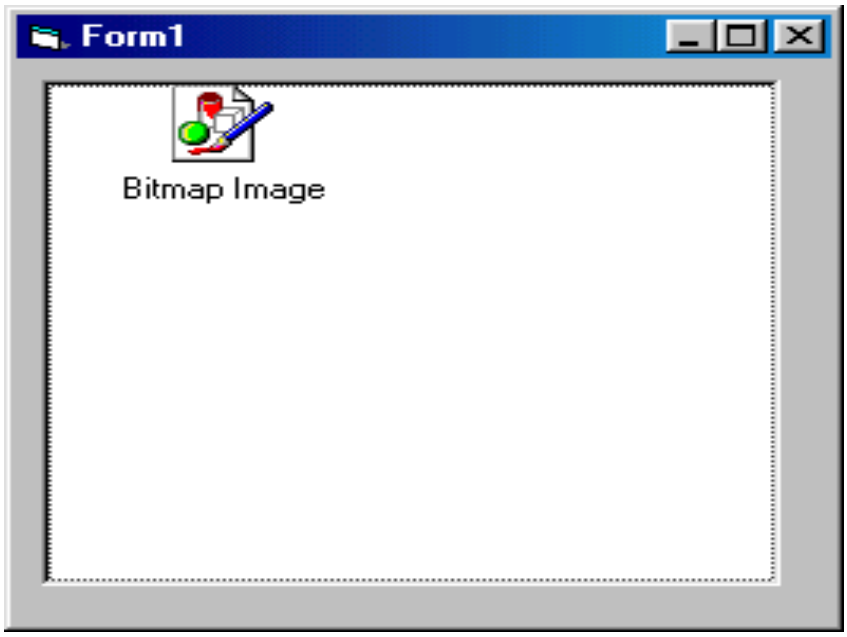
1. Viết chương trình cho phép nhập vào bán kính r của một hình tròn. Tính chu vi và diện tích của hình tròn theo công thức :

Chu vi CV = 2*Pi*r

Diện tích Dt = Pi*r*r

Hiển thị các kết quả lên màn hình.

4) Thiết kế chương trình có giao diện như hình dưới và thực hiện các chức năng sau:



Hình I.9: Lựa chọn tên

- Mỗi khi người sử dụng chương trình nhập thông tin vào 2 ô TextBox, sau đó nhấp chọn nút Thêm, giá trị của ô Mã số được đưa vào ComboBox, còn giá trị của ô Họ và tên được đưa vào ListBox.
- Mỗi khi họ chọn một mã số nào đó trong ComboBox, giá trị họ và tên tương ứng cũng sẽ được chọn trong ListBox; đồng thời chúng sẽ được hiển thị lên trên các điều khiển TextBox tương ứng (như hình). (Xử lý sự kiện Combo1_Click & List1_Click)
- Đối với mã số và họ tên của một người, ta có thể sửa đổi giá trị của chúng trong các ô nhập TextBox, sau đó chọn nút Sửa, giá trị của chúng trong ComboBox & ListBox cũng sửa đổi theo.
- Khi người dùng chọn một mã số (hay họ tên) trên ComboBox (hoặc ListBox), sau đó họ chọn Xóa, các thông tin này được xóa ra khỏi ComboBox & ListBox.

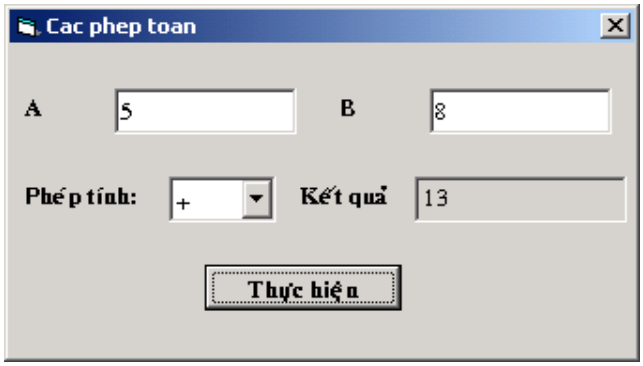
CÁC CẤU TRÚC LẬP TRÌNH TRONG VB

Bài tập có hướng dẫn

Bài tập 1II-1

SỬ DỤNG SELECT CASE

Tạo thư mục Basic\Bt1II-1. Thiết kế chương trình có giao diện & lưu trong thư mục trên:



Hình I.10: Select Case

Ở đây, người sử dụng chương trình nhập vào một tuổi nào đó trong ô nhập tuổi, sau đó họ nhấp nút Nhận xét, một nhận xét sẽ xuất hiện ứng với tuổi mà họ nhập từ bàn phím.

Lúc này ta sử dụng toán tử so sánh (=, <, <=, >, >=, <>) cùng với các từ khóa Is và To trong biểu thức.

Is: so sánh biến với biểu thức được liệt kê sau từ khóa Is.

To: định nghĩa phạm vi của giá trị.

Sự kiện Command1_Click():

```
Dim Age As Integer

Age = Val(Text1.Text)

Select Case Age
```

```
Case Is < 18

Label2.Caption = "Ban con thieu nien, ban phai hoc thoi!"

Case 18 To 30

Label2.Caption = "Ban da truong thanh, lap gia dinh thoi!"

Case 31 To 60

Label2.Caption = "Lua tuoi trung nien roi!"

Case Else

Label2.Caption = "Ban co con chau day dan roi nhe!"

End Select
```

Bài tập 1II-2

BIẾN VÀ CẤU TRÚC

Bước 1: Tạo thư mục Basic\Bt1II-2. Tạo dự án mới (VB Standard EXE) trong thư mục trên; thêm một modul vào dự án, trong modul nay thêm vào đoạn mã sau:

```
Public Const tieude As String = "Quan ly hanh chinh"

Public Const sohieu As String = "1.0"
```

Thêm đoạn mã sau vào hàm xử lý sự kiện Form_Load của Form1:

```
Form1.Caption = tieude & " phien ban " & sohieu
```

Chạy ứng dụng, ta thấy tiêu đề của Form: “Quan ly hanh chinh phien ban 1.0”.

Bây giờ, mở Modul1 và thay Public bằng Private. Chạy chương trình. Điều gì xảy ra?

Bước 2: Đổi các khai báo trên thành Public, thêm dòng sau đây vào đầu thủ tục Form_Load:

tieude = “Loi xuat hien” & “Hang so khong the thay doi duoc.”

Chạy chương trình, điều gì xảy ra?

Bước 3: Thêm dòng sau trong hàm xử lý sự kiện Form_Resize:

MsgBox “FORM RESIZE”

Bước 4: Chạy chương trình, khi Form bắt đầu được hiển thị (sự kiện Form_Load), sự kiện Resize của Form được thực hiện. Chỉ có hàm xử lý sự kiện Resize mới cho biết chắc rằng hàm Form_Load được thực thi. Để kiểm chứng ta tạo một biến trên form và trong hàm Form_Load ta thiết lập giá trị của nó. Sau đó, hàm Form_Resize có thể kiểm tra biến và xử lý trên biến này.

Bước 5: Khai báo một biến Private trong Form1 tên sukienLoad:

Private sukienLoad As Boolean

Trong hàm Form_Load, đặt giá trị True cho biến trên:

sukienLoad = True

Bây giờ ta kiểm tra giá trị của biến trong hàm Form_Resize. Thêm vào đoạn mã sau trong hàm Form_Resize:

If sukienLoad = True Then

SukienLoad =False

Exit Sub

End If

MsgBox “Form Resize”

Chạy ứng dụng, khi Form bắt đầu được hiển thị, ta không thấy xuất hiện câu thông báo, nhưng khi ta thay đổi kích thước của Form (nhấn các nút _, của form), câu thông báo lại xuất hiện. Ở đây ta đã sử dụng một biến làm trung gian cho sự giao tiếp giữa sự kiện Form_Load và sự kiện Form_Resize. Bởi vì cả 2 hàm này nằm trong Form1, nên ta có khai báo Private cho chúng, các ứng dụng khác không thể truy xuất đến các biến này.

CHƯƠNG TRÌNH CON

Bước 6: Ta viết một chương trình con để xử lý chuỗi. Đầu vào của chương trình con là một chuỗi, kết quả của chương trình con là chuỗi đó nhưng các từ đều được viết hoa ký tự đầu tiên. Bài tập này giúp ta khai báo (định nghĩa) một chương trình con và gọi thực thi chương trình con đó trong chương trình ứng dụng của mình.

Chọn Modul1 trong cửa sổ soạn thảo chương trình, sau đó nhấp chọn Tools\Add Procedure. Định nghĩa một hàm public tên Doihoa() như sau:

Public Function Doihoa(s As String) As String

Dim s1 As String

Dim s2 As String

Do While InStr(s, " ") <> 0

s1 = Left(s, InStr(s, " "))

s = Right(s, Len(s) - InStr(s, " "))

' Doi chu hoa

```
s1 = UCase(Left(s1, 1)) & Right(s1, Len(s1) - 1)

s2 = s2 & s1

Loop

' Tra ket qua

Doihoa = s2 & " " & s

End Function

' Ham nay khong viet hoa tu cuoi cung.
```

Bước 7: Hàm Doihoa có nhiệm vụ nhận vào một chuỗi và đổi ký tự đầu tiên của các từ trong chuỗi thành chữ hoa. Bây giờ ta kiểm tra hàm này như sau: Thêm một TextBox và một nút nhấn (Button) lên Form1. Nhấp vào Button, ta thêm đoạn mã sau vào hàm xử lý sự kiện Command1_Click:

```
Form1.Caption = Doihoa(Text1.Text)
```

Chạy ứng dụng, nhập một chuỗi vào Text1, nhấp Command1. Chuỗi chữ hoa sẽ xuất hiện trên tiêu đề của Form1.

Bước 8: Sửa lại sao cho có thể viết hoa ký tự đầu tiên của tất các từ.

Bài tập 1II-3

LỰA CHỌN VỚI LISTBOX

Bước 1: Tạo thư mục Basic\Bt1II-3. Tạo dự án mới VB Standard EXE trong thư mục trên, sau đó tạo Form có dạng sau:

Hình I.11: Lựa chọn với ListBox

Ta có 2 ListBox và các nút nhấn (Button); trong đó:

Nút > chuyển một phần tử từ trái sang phải

Nút < chuyển một phần tử từ phải sang trái.

Nút >> chuyển tất cả các phần tử từ trái sang phải.

Nút << chuyển tất cả các phần tử từ phải sang trái.

Thêm 2 ListBox và 4 Button vào Form1. Trong hàm xử lý sự kiện Form_Load thêm vào đoạn mã:

```
List1.AddItem "Thing 1"

List1.AddItem "Thing 2"

List1.AddItem "Thing 3"

List1.AddItem "Thing 4"

List1.AddItem "Thing 5"

List1.AddItem "Thing 6"
```

Chạy chương trình.

Bước 2: Thêm hàm xử lý sự kiện Click cho nút nhấn 1 (>) Command1_Click:

```
' Kiem tra co chon hay khong?

If Form1.List1.ListIndex = -1 Then Exit Sub

' Chep tu trai sang phai

Form1.List2.AddItem Form1.List1.List(Form1.List1.ListIndex)

' Xoa ben trai

Form1.List1.RemoveItem Form1.List1.ListIndex
```

Bước 3: Chạy chương trình, chọn phần tử trong List1 và nhấp nút >, phần tử đó chuyển sang List2. Bây giờ ta làm ngược lại: chuyển phần tử được chọn từ List2 sang List1. Trở về cửa sổ soạn thảo; chọn đoạn mã vừa nhập trong List1, chọn Edit\Copy trong menu của VB. Nhấp lên Button <, chọn Edit\Paste. Bây giờ ta sửa lại đoạn mã sau trong hàm xử lý sự kiện Command2_Click:

```
' Kiem tra co chon hay khong?

If Form1.List2.ListIndex = -1 Then Exit Sub

' Chep tu phai sang trai

Form1.List1.AddItem Form1.List2.List(Form1.List2.ListIndex)

' Xoa ben phai

Form1.List2.RemoveItem Form1.List2.ListIndex
```

Bước 4: Lưu dự án và chạy chương trình.

Ta nhận thấy 2 đoạn mã lệnh trên (cho Button < và >) là như nhau (chỉ đổi chỗ List1 cho List2 và ngược lại). Do đó ta sẽ viết một chương trình con để chuyển dữ liệu từ ListBox này sang ListBox kia, và trong hàm xử lý sự kiện của 2 Button ta chỉ cần gọi chương trình con này để chuyển dữ liệu.

Thêm một Modul mới vào dự án tên Modul1, chọn Tool\Add Procedure để thêm một chương trình con vào tên Chuyendulieu()

Vào Modul1, sửa đổi lại thủ tục chuyển dữ liệu như sau:

```
Public Sub Chuyendulieu(L1 As ListBox, L2 As ListBox)

' Kiem tra co chon hay khong?

If L1.ListIndex = -1 Then Exit Sub

' Chep

L2.AddItem L1.List(L1.ListIndex)

' Xoa ben trai

L1.RemoveItem L1.ListIndex

End Sub
```

Bây giờ hàm xử lý sự kiện của Command1 (Command1_Click) ta sửa lại như sau:

```
Call Chuyendulieu(Form1.List1, Form1.List2)
```

Hàm Command2_Click:

```
Call Chuyendulieu(Form1.List2, Form1.List1)
```

Lưu dự án và chạy chương trình. Kiểm tra kết quả.

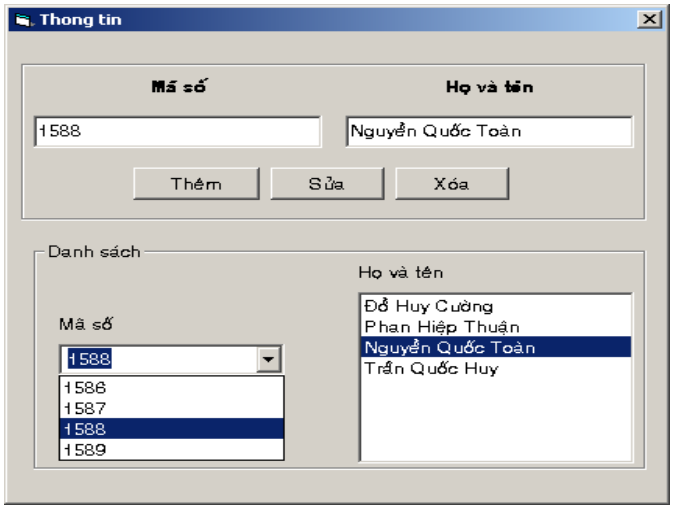
Bài tập 1I -4

TRUYỀN THEO ĐỊA CHỈ VÀ TRUYỀN THEO GIÁ TRỊ

Bước 1: Tham số đưa vào chương trình con được truyền theo một trong 2 cách: theo địa chỉ và theo giá trị. Bây giờ ta tạo dự án mới trong thư mục Basic\Bt2-4 để kiểm tra chúng.

Bước 2: Tạo Form1 như sau:

Hình I.12: Truyền tham số321



Button 1: Name: cmdTTri; Caption: Truyen tri

Button 2: Name: cmdTChieu; Caption: Tham chieu

TextBox: Name: Text1

Label 1: Name: lblTruoc

Label 2: Name: lblTrong

Label 3: Name: lblSau

Bước 3: Thêm 1 Modul vào dự án tên là Modul1, chọn Tools\Add Procedure thêm thủ tục Thamchieu như sau:

Name: Thamchieu

Type: Sub

Scope: Public

Bước 4: Thêm đoạn mã sau trong thủ tục Thamchieu

```
Public Sub Thamchieu(so As Integer)

so = so + 2

Form1.lblTrong.Caption = Str(so)

End Sub
```

Bước 5: Chọn Tool\Add Procedure để thêm thủ tục Truyentri như sau:

Name: Truyentri

Type: Sub

Scope: Public

Bước 6: Thêm đoạn mã sau trong thủ tục Truyentri

```
Public Sub Truyentri(ByVal so As Integer)

so = so + 2

Form1.lblTrong.Caption = Str(so)

End Sub
```

Bước 7: Sự khác nhau giữa 2 thủ tục trên là từ khóa ByVal trong thủ tục Truyentri. Bây giờ ta thêm thủ tục xử lý biến cố Click của Button

cmdTTri. Thêm đoạn mã sau:

```
Private Sub cmdTTri_Click()

Dim n As Integer

n = Val(Text1.Text)

lblTruoc.Caption = Str(n)

Call Truyentri(n)

lblSau.Caption = Str(n)

End Sub
```

Bước 9: Thêm hàm xử lý biến cố cmdTChieu_Click cho Button cmdTChieu:

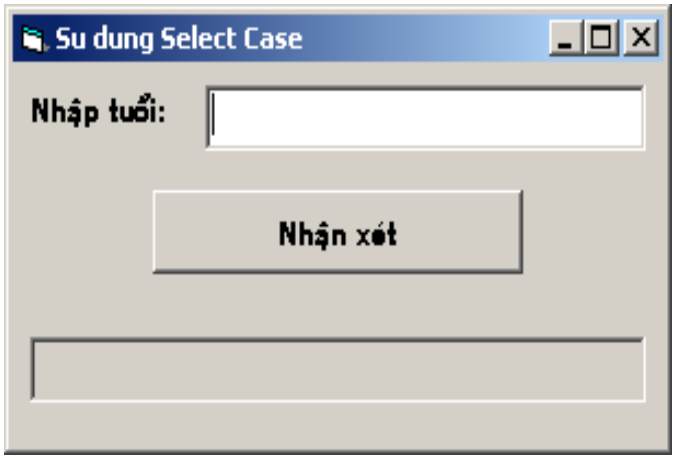
```
Private Sub cmdTChieu_Click()

Dim n As Integer

n = Val(Text1.Text)

lblTruoc.Caption = Str(n)

Call Thamchieu(n)

lblSau.Caption = Str(n)

End Sub
```

Bước 10: Lưu dự án, chạy chương trình. Nhập số bất kỳ vào ô TextBox rồi nhấp nút Truyen tri, sau đó nhấp nút Tham chieu. Kiểm tra kết quả. Giải thích.

THAM SỐ TÙY CHỌN

Bước 1: Tạo thư mục tên Basic\Bt1II-5. Tạo dự án mới trong thư mục này.

Bước 2: Tạo Form như sau:



Hình I.13: Lấy thời gian

Trong đó:

Label: Name: lblTg

Button 1: Name: cmdGioPhut

Button 2: Name: cmdGioPhutGiay

Bước 3: Thêm modul mới vào dự án tên Modul1. Chọn Tool\ Add Procedure tạo thủ tục:

Name: Laythoigian

Type: Sub

Scope: Public

Bước 3: Thêm đoạn mã sau vào thủ tục trên:

```
Public Sub Laythoigian(gio As String, phut As String, Optional giay As String)

' Tham so thu 3 co tu khoa Optional, nghia la ta

' co the goi thu tuc co the co tham so nay hay khong co deu duoc

' Ham IsMissing kiem tra xem tham so nay co hay khong

If IsMissing(giay) Then giay = ""

Dim hientai

hientai = Now

gio = Format$(hientai, "hh")

phut = Format$(hientai, "nn")

giay = Format$(hientai, "ss")

End Sub
```

Bước 4: Thêm thủ tục xử lý sự kiện cho Button cmdGiophutgiay, trong thủ tục này chèn đoạn mã sau:

```
Private Sub cmdGiophutgiay_Click()

Dim gioht As String

Dim phutht As String

Dim giayht As String

Call Laythoigian(gioht, phutht, giayht)

lblTg.Caption = gioht & ":" & phutht & ":" & giayht
```

End Sub

Bước 5: Thêm thủ tục xử lý sự kiên cho Button cmdGiophut, trong thủ tục này chèn đoạn mã sau:

Private Sub cmdGiophut_Click()

Dim gioht As String

Dim phutht As String

' Khong su dung tham so thu ba

Call Laythoigian(gioht, phutht)

lblTg.Caption = gioht & ":" & phutht

End Sub

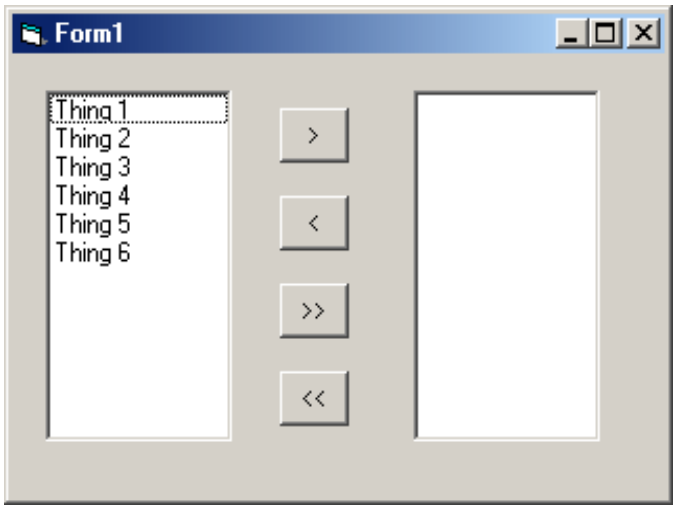
Bước 6: Lưu dự án lại và chạy chương trình. Kiểm tra kết quả.

Bài tập 1II-6

PARAM ARRAY

Bước 1: Tạo thư mục Basic\Bt1II-6. Tạo dự án mới trong thư mục này.

Bước 2: Tạo Form như hình sau:



Hình I.14: Param Array

Trong đó:

ListBox: Name: lstTen

Button: Name: cmdds; Caption: Them vao danh sach

Bước 3: Chèn modul mới vào dự án tên Modul1. Sau đó, chọn Tool\Add Procedure để chèn thủ tục sau:

Name: Diends

Type: Sub

Scope: Public

Bước 3: Chèn đoạn mã sau vào thủ tục Diends

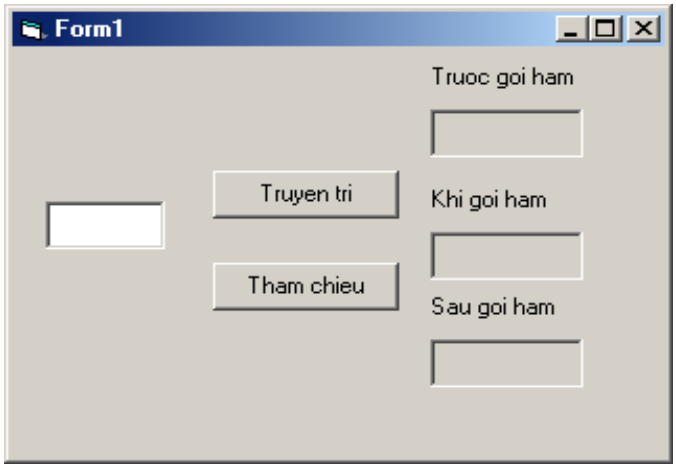
```
Public Sub Diends(ParamArray Ten() As Variant)

' Su dung ParamArray thi mang phai kieu Variant va

' mang nay la tham so cuoi cung cua thu tuc

Dim hten As Variant
```

```
For Each hten In Ten()

Form1.lstTen.AddItem hten

Next

End Sub
```

Bước 4: ParamArray cho phép không cần xác định số lượng các đối số trong một chương trình con. Bây giờ, thêm hàm xử lý sự kiện cho nút cmdds: cmdds_Click:

```
Private Sub cmdds_Click()

Call Diends("Huynh Xuan Hiep", "Nguyen Van Linh", "Lam Hoai Bao")

Call Diends

Call Diends("Phan Huy Cuong")

End Sub
```

Bước 5: Lưu dự án lại và chạy chương trình. Kiểm tra kết quả (hình bên dưới). Lưu ý đến lời gọi thủ tục trong sự kiện cmdds_Click (số lượng đối số khác nhau)



Hình I.15: Kết quả Param Array

Bài tập 1II-7

XỬ LÝ CHUỖI

[missing_resource: .png]

12345Bước 1: Tạo dự án mới trong thư mục Basic\Bt2-7 với giao diện như sau:

Hình I.16: Xử lý chuỗi

1: Form: Name: frmMain; MinButton: False; MaxButton: False; Font: VNI-Times.

2: Label: Name: lblTen.

3: TextBox: Name: txtTen.

4: CommandButton: Name: cmdTen; Caption: Tách tên.

5: CommandButton: Name: cmdCKT; Caption: Cắt khoảng trắng.

Bước 2: Tạo một hàm cắt khoảng trắng như sau:

```
Private Function ATrim(ByVal Name As String) As String

Name = LTrim(RTrim(Name))

Do While InStr(Name, "  ") <> 0

Name = Replace(Name, "  ", " ")

Loop
```

```
ATrim = Name

End Function
```

Bước 3: Trong cửa sổ thiết kế Form; nhấp đúp vào Tách tên, ta xử lý đoạn mã cho sự kiện này:

```
Private Sub cmdTen_Click()

Dim sName As String, Name As String

sName = ATrim(StrConv(txtTen.Text, vbProperCase))

Dim i As Long

i = InStrRev(sName, "  ")

Name = Right(sName, Len(sName) - i)

MsgBox Name & ": " & Str(Len(Name))

End Sub
```

Bước 4: Sau đó, trở lại cửa sổ thiết kế, nhấp đúp vào Cắt khoảng trắng, ta xử lý:

```
Private Sub cmdCKT_Click()

Dim sName As String

sName = ATrim(StrConv(txtTen.Text, vbProperCase))

MsgBox sName, , "Kieu du lieu chuoi"

End Sub
```

Bước 5: Lưu dự án và chạy chương trình.

Bài tập 1II-8

XỬ LÝ LỖI

Bước 1: Tạo một dự án mới. Dùng Tools\Add Procedure thêm một thủ tục mới tên GoiThuTuc vào Form1 với nội dung như sau:

```
Public Sub GoiThuTuc()

Dim bien As Integer

MsgBox "Truoc khi gan tri cho bien"

bien = "Bien nguyen khong nhan gia tri la chuoi"

MsgBox "Sau khi gan tri cho bien: " & "Bien = " & Format(bien)

End Sub
```

Bước 2: Thủ tục xử lý sự kiện Form_Load có nội dung như sau:

```
Private Sub Form_Load()

MsgBox "Truoc khi goi thu tuc"

Call GoiThuTuc

MsgBox "Sau khi goi thu tuc"

End Sub
```

Lưu dự án vào thư mục Basic\Bt1II-8:

Form: tên là form1

Project: Debug

Bước 3: Chạy chương trình. VB đưa ra hộp thoại để bắt lỗi (debug) chương trình. Ta chọn End để trở về cửa sổ soạn thảo.

Tạo tập tin thực thi tên Debug.exe bằng cách chọn File\Make Debug.exe. Chạy tập tin Debug.exe từ Windows Explorer ta nhận được hộp thoại báo lỗi và chương trình tự động chấm dứt.

Nhận xét kết quả khi thực hiện chương trình.

Bước 4: Bây giờ ta thêm vào đoạn mã xử lý lỗi trong thủ tục của sự kiện Form_Load:

```
Private Sub Form_Load()

On Error GoTo Xulyloi

MsgBox "Truoc khi goi thu tuc"

Call GoiThuTuc

MsgBox "Sau khi goi thu tuc"

Thoat:

Exit Sub

Xulyloi:

MsgBox "Su kien Form_Load - Loi xay ra: " & Err.Description

Resume Thoat

End Sub
```

Bước 5: Lưu dự án và chạy chương trình. Nhận thấy, thay vì ta nhận được câu thông báo lỗi từ VB, một hộp thoại báo lỗi do ta đưa vào xuất hiện. Lưu ý, những lỗi được bắt trong thủ tục Form_Load (chứ không phải trong GoiThuTuc()). Nguyên nhân vì thủ tục GoiThuTuc() được gọi bởi thủ tục xử lý sự kiện Form_Load.

Bước 6: Biên dịch lại thành tập tin Debug.exe, chạy nó. Nhận xét kết quả.

Bước 7: Các kết quả trên cho ta biết được các lỗi trong sự kiện Form_Load được xử lý bởi các thao tác bắt lỗi trong thủ tục Form_Load. Nhưng nếu thủ tục GoiThuTuc() cũng có các thao tác bắt lỗi chương trình thì sao? Đơn giản giả sử một lỗi xuất hiện trong GoiThuTuc(). Bộ phận xử lý lỗi của GoiThuTuc (do ta thêm vào để bắt lỗi chương trình) sẽ thực thi thay vì đoạn lệnh bắt lỗi của sự kiện Form_Load được thực hiện. Khi GoiThuTuc chấm dứt, quyền xử lý lỗi mới trao lại cho sự kiện Form_Load.

Sửa lại thủ tục GoiThuTuc như sau:

```
Public Sub GoiThuTuc()

Dim bien As Integer

On Error GoTo Xulyloicucbo

MsgBox "Truoc khi gan tri cho bien"

bien = "Bien nguyen khong nhan gia tri la chuoi"

MsgBox "Sau khi gan tri cho bien: " & "Bien = " & Format(bien)

Thoatthutuc:

Exit Sub

Xulyloicucbo:

MsgBox "GoiThuTuc() - Loi xay ra: " & Err.Description

Resume Thoatthutuc

End Sub
```

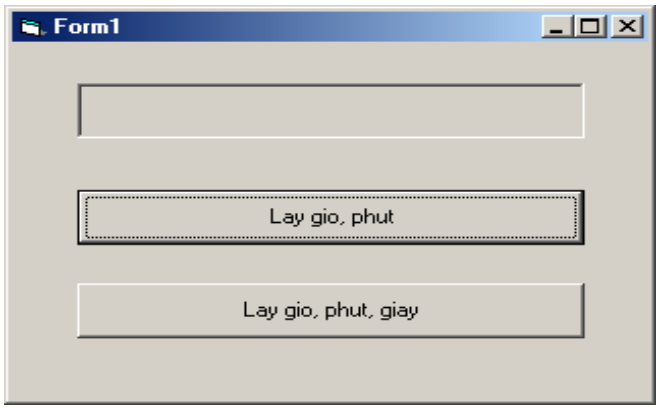
Bước 8: Lưu dự án và chạy chương trình. Thay đoạn mã Resume Thoatthutuc bằng Resume và chạy chương trình. Một vòng lặp vô tận xảy ra do chương trình sẽ quay lại đoạn mã bị lỗi và cố gắng thực thi nó; để thoát chương trình ta phải bấm tổ hợp phím Ctrl + Break.

Bây giờ thay Resume bằng Resume Next và chạy lại chương trình. Nhận xét kết quả. Giải thích.

Bài tập tự làm

1. Thiết kế chương trình cho phép nhập vào các hệ số a, b của phương trình bậc 1 dạng: ax+b=0; sau đó giải phương trình này. Giao diện chương trình có thể như sau:

Hình I.17: Phương trình bậc 1



1. Thiết kế chương trình cho phép nhập vào các hệ số a, b, c của phương trình bậc 2 dạng: ax2 + bx + c=0; sau đó giải phương trình này.
2. Thiết kế chương trình cho phép nhập vào một ký tự, sau đó kiểm tra xem ký tự đó thuộc tập hợp nào trong các tập ký tự sau:

Các ký tự chữ hoa: 'A' ...'Z'

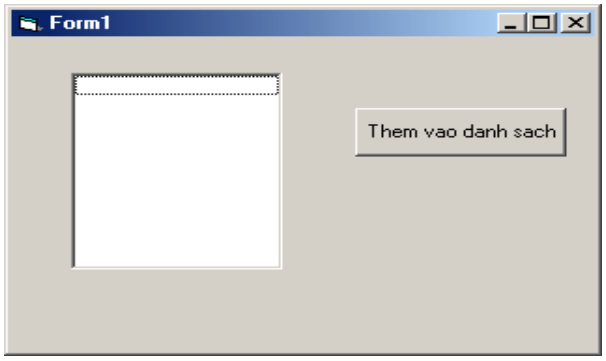
Các ký tự chữ thường: 'a' ... 'z'

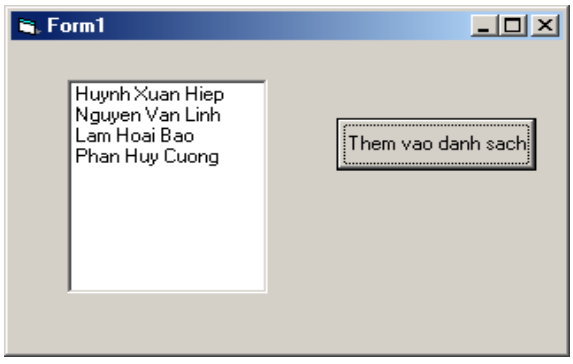
Các ký tự chữ số : '0' ... '9'

Các ký tự khác.

1. Giải phương trình bậc 1 bằng cách sử dụng cấu trúc Select Case
2. Tạo một chương trình hiển thị một danh sách chọn lựa cho người dùng trong một ListBox, sau đó xử lý với cấu trúc quyết định Select Case.

Mục đích của điều khiển sự kiện này là hiển thi một danh sách các quốc gia, sau đó hiển thị một thông điệp chào mừng bằng ngôn ngữ bản xứ khi người dùng chọn quốc gia của họ.





Hình I.18: Lời chào các nước

Chẳng hạn: Tiếng Anh: Hello, programmer

Tiếng Đức: Hallo, programmierer

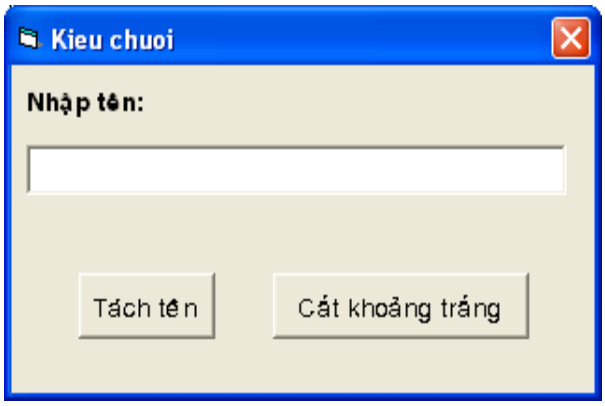
Tiếng Tây Ban Nha: Hola, programador

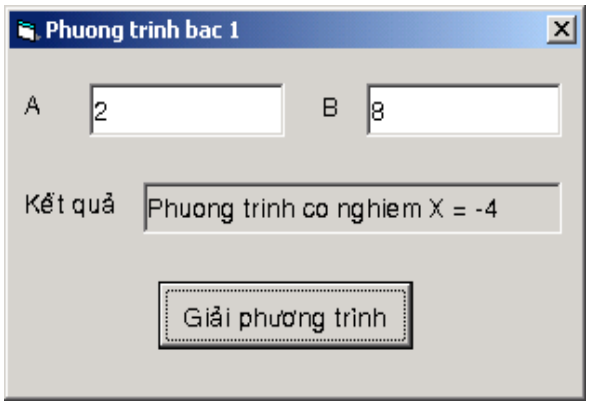
Tiếng Ý: Ciao, programmatori

1. Sử dụng vòng lặp For.. Next

Sử dụng For.. Next để thay đổi độ lớn ký tự trên một Form bằng cách thay đổi thuộc tính FontSize của Form.

Thiết kế Form có giao diện:





Hình I.19: For…Next

Sự kiện Command1_Click()

Dim i As Integer

For i = 1 To 10

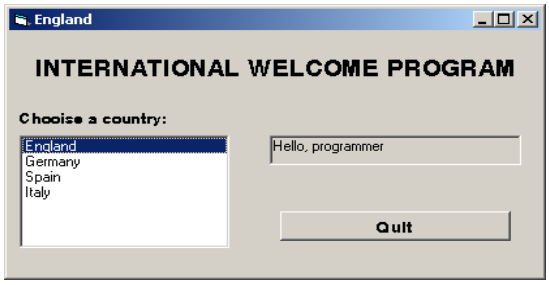
Form1.FontSize = 10 + i

Print "Line "; i

Next

Chạy chương trình.

1. Thiết kế chương trình cho phép tính N! (N! = 1*2*3*…*N). Giao diện đề nghị:



Hình I.20: Tính N!

1. Thiết kế chương trình cho phép nhập vào một số nguyên N; sau đó tính các tổng sau:

a. S=1 + 2 + ... + n

b. S=1/2 + 2/3 + ... +n/(n+1)

c. S= - 1 +2 - 3 +4 - ... + (-1)nn

1. Thiết kế chương trình cho phép nhập vào số nguyên dương N; sau đó tìm số nguyên dương k nhỏ nhất sao cho $\frac{2}{1*3} + \frac{3}{2*4} + ... + \frac{k}{(k-1)*(k+1)} \geq$N.
2. Thiết kế chương trình cho phép nhập vào 2 số nguyên A, B; sau đó tìm UCLN và BCNN của hai số a và b theo thuật toán sau (Ký hiệu UCLN của a, b là (a,b) còn BCNN là [a,b])

- Nếu a chia hết cho b thì (a,b) = b

- Nếu a = b*q + r thì (a,b) = (b,r)

- [a,b] = a*b/(b,r)

1. Thiết kế chương trình cho phép nhập vào số nguyên N; sau đó viết 1 hàm tính N!; cuối cùng hiển thị kết quả giá trị N!.
2. Thiết kế chương trình cho phép nhập vào 2 số nguyên N, K; sử dụng hàm tính N! ở trên, viết một hàm tính giá trị tổ hợp chập K của N phần tử theo công thức $C_N^K = \frac{N!}{K!*(N-K)!}$.
3. Thiết kế chương trình cho phép nhập vào số thực X và số nguyên N; sau đó viết các hàm tính các tổng sau rồi hiển thị kết quả:

- S= 1 + x +x2 + x3 + ... + xn

- S= 1 - x +x2 - x3 + ... (-1)n xn

- S= 1 + x/1! +x2/2! + x3/3! + ... + xn/n!

1. Sử dụng vòng lặp Do While … Loop thiết kế chương trình cho phép nhập vào một số nguyên, sau đó thông báo kết quả xem số đó có phải là số nguyên tố hay không?

Đoạn chương trình kiểm tra số nguyên N có nguyên tố hay không:

i = 2

Do While (i < N) And (N Mod i <> 0)

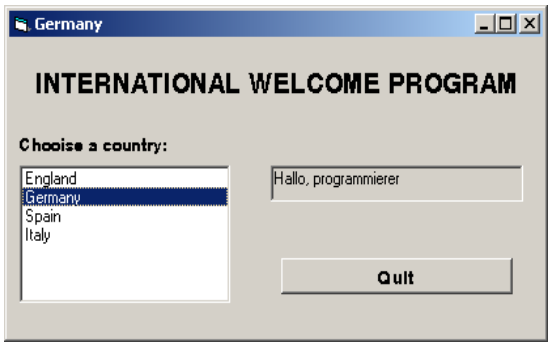
i = i + 1

Loop

If i = N Then N là số nguyên tố

Else N không là nguyên tố

1. Làm lại bài tập 11 (tính N!) nhưng sử dụng vòng lặp Do While … Loop.
2. Làm lại bài tập 15 (kiểm tra số nguyên tố) nhưng bằng cách sử dụng Do Until … Loop.
3. Làm lại bài tập 11 (tính N!) nhưng sử dụng vòng lặp Do Until … Loop.
4. Thiết kế chương trình cho phép nhập vào một số nguyên N; sau đó phân tích số nguyên này ra thừa số nguyên tố. Giao diện chương trình có thể như sau:



Hình I.21: Thừa số nguyên tố

1. Sử dụng điều khiển định thời (Timer).

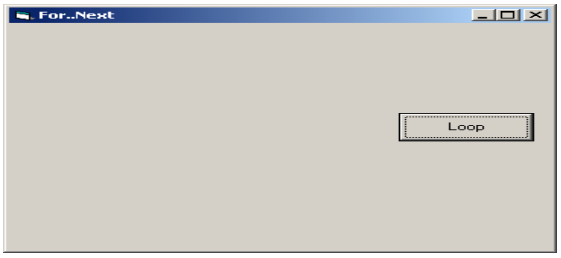
Tạo một chương trình cho phép người dùng 15 giây để nhập mật khẩu trong một TextBox.

Nếu người dùng không nhập mật khẩu đúng trong thời gian nói trên, chương trình hiển thị thông báo “Time Expired” (Hết thời gian) và đóng

chương trình.

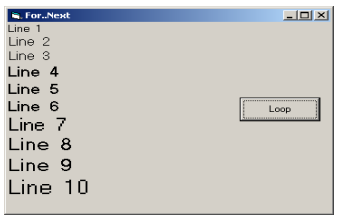
Thời gian làm bài tập: 30 phút.

Giao diện đề nghị:



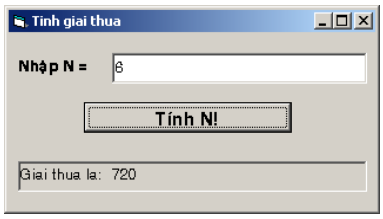
Hình I.22: Giao diện

- Nhập vào mật khẩu cần thiết (giả sử mật khẩu là: Secret)
- Nếu nhập đúng mật khẩu, rồi nhầp nút Nhập, một hộp thông báo xuất hiện với nội dụng: Ban dang nhap thanh cong.



Hình I.23: Lỗi đăng nhập

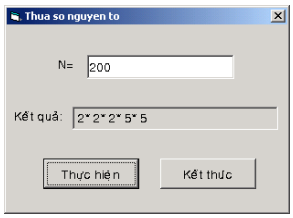
- Nếu nhập mật khẩu sai, rồi nhấp nút Nhập, một thông báo xuất hiện với nội dung: Xin loi, chung toi khong biet ban!



Hình I.24: Lỗi đăng nhập

Sau đó nhấp nút OK trên hộp thông báo này thì chương trình cho bạn nhập lại mật khẩu.

- Nếu thời gian quá 15 giây mà người dùng chưa nhập đúng mật khẩu thì một thông báo sẽ hiện lên Xin loi, thoi gian da het; sau đó chương trình sẽ kết thúc.



Hình I.25: Báo hết giờ

1. Thiết kế chương trình tương tự như ứng dụng Canculator của Windows.

Lập trình xử lý giao diện & đồ họa
Mục tiêu: Chương này giới thiệu về cách tạo menu cũng như một số hàm xử lý đồ họa, những thành phần quan trọng trong các ứng dụng chạy trên Windows.

Tập tin

Mục tiêu: Chương này nhằm mục đích rèn luyện sinh viên các kỹ năng thao tác với hệ thống tập tin của Windows trong VB. Bên cạnh đó, việc hệ thống lại các kiến thức của các chương trước cũng là một mục tiêu quan trọng của chương.

Học xong chương này, sinh viên phải nắm được các vấn đề sau:

- Sử dụng mô hình hệ thống tập tin.

- Cách thức truy cập tập tin tuần tự.

- Cách thức truy cập tập tin truy xuất ngẫu nhiên.

Kiến thức có liên quan:

Giáo trình Visual Basic, Chương 7.

Tài liệu tham khảo:

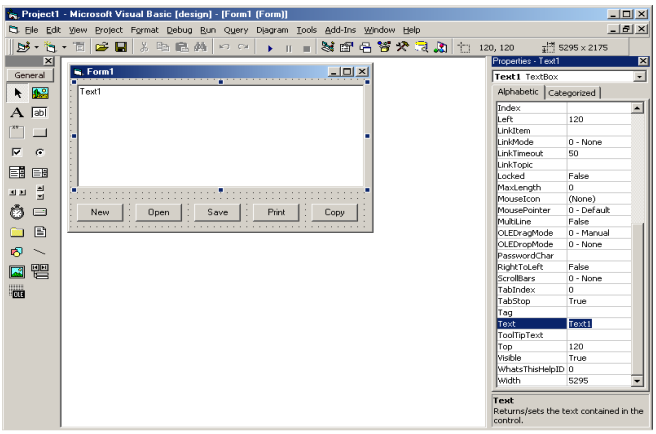
- Visual Basic 6 Certification Exam Guide - Chapter 7, Page 191; Chapter 13, Page 377 - Dan Mezick & Scot Hillier - McGraw-Hill - 1998.

BÀI TẬP HƯỚNG DẪN

Bài tập 3-1:

XUẤT NHẬP TẬP TIN VĂN BẢN

Bước 1: Tạo Project mới tên Bt3-1 trong thư mục Basic\Bt3-1. Tạo giao diện có dạng sau:



Hình III.1: Tập tin văn bản

Item 1 – TextBox

NameText1

Height2220

Width6630

MultilineTrue

ScrollBarsBoth

Item 2 – CommandButton

NameCommand1

CaptionNew

Item 3 – CommandButton

NameCommand2

CaptionOpen

Item 4 – CommandButton

NameCommand3

CaptionSave

Item 5 – CommandButton

NameCommand4

CaptionPrint

Item 6 – CommandButton

NameCommand5

CaptionCopy

Bước 2: Nút New có nhiệm vụ xóa văn bản trong TextBox để ta có thể khởi tạo một tài liệu mới. Do đó, trong hàm sự kiện Command1_Click, thêm vào đoạn mã:

Text1.Text = “”

GHI CHUỖI LÊN TẬP TIN

Bước 3: Ở đây ta nhập vào đoạn văn bản rồi ghi lên tập tin. Để đơn giản ta đọc và ghi từ một tập tin văn bản duy nhất tên là vidu.txt nằm trong thư mục của dự án của mình (ở đây là thư mục Bt5-1). Để ghi lên tập tin, trong hàm sự kiện Command3_Click, thêm đoạn mã sau:

```
' Ghi len tap tin

Open App.Path & "\vidu.txt" For Output As #1

' Ghi du lieu

Print #1, Text1.Text

' Dong tap tin

Close #1
```

MsgBox "Van ban da duoc luu"

Bước 4: Việc thao tác trên tập tin được thực hiện nhờ thẻ tập tin. Thực chất đây là một số nguyên chỉ bởi VB một liên kết đến một tập tin xác định để xuất hay nhập vào tập tin đó. Ở đây là sử dụng #1. Câu lệnh Print sử dụng thẻ tập tin để ghi văn bản lên tập tin. Khi việc ghi hoàn tất, thẻ tập tin được đóng lại nhờ câu lệnh Close.

Bước 5: Chạy ứng dụng, nhấp nút Command3. Nếu chương trình thực thi tốt, ta có thể mở tập tin vidu.txt trong Notepad xem.

ĐỌC TỪ TẬP TIN VĂN BẢN

Bước 6: Đọc tập tin từ đĩa tương tự như ghi tập tin. Chèn đoạn mã sau trong hàm xử lý sự kiện Command2_Click

```
Text1.Text = ""

Close #1

' Mo tap tin

Open App.Path & "\vidu.txt" For Input As #1

Dim filetext As String ' Bien chuoi luu van ban

Do While Not EOF(1)

Input #1, filetext ' Doc tung dong

' Hien thi trong TextBox, chu y them vao ky tu xuong dong

Text1.Text = Text1.Text & filetext & vbCrLf

Loop

Close #1
```

Bước 7: Chạy ứng dụng. Nhấp nút Command2 để đọc từ tập tin vidu.txt vào TextBox. Ở đây ta có định nghĩa một biến trong lệnh

Dim filetext as String

Ở đây, mỗi lần ta đọc từng dòng trong tập tin vidu.txt; mỗi lần đọc như vậy ta lưu vào biến kiểu chuỗi filetext; sau đó ta nối chuỗi filetext vào sau chuỗi Text1.Text (hiển thị trong TextBox). Quá trình trên được thực hiện liên tục đến khi đọc hết nội dung tập tin nhờ vào vòng lặp:

Do While Not EOF(1)

EOF là một hàm được định nghĩa sẵn trong VB, hàm này có nhiệm vụ kiểm tra xem có đạt đến cuối tập tin hay không? Nếu nội dung tập tin vẫn chưa được đọc hết, quá trình đọc vẫn tiếp tục đến khi EOF là True.

Input #1

Đọc một chuỗi từ tập tin cho đến khi gặp ký tự xuống dòng. Ký tự xuống dòng này được bỏ qua trong lệnh Input; do đó nếu muốn hiển thị thành nhiều dòng trên TextBox, ta phải thêm vào ký tự xuống dòng cho mỗi dòng ta đọc được từ tập tin sau đó ta mới hiển thị trên TextBox. Hằng số vbCrLf là sự liên kết 2 ký tự xuống dòng và về đầu dòng.

IN VĂN BẢN RA MÁY IN

Bước 8: Nếu máy in được nối vào, máy in phải được kích hoạt. Ta có thể kiểm tra chúng bằng cách in thử vài dòng văn bản trong Word hay trong Notepad.

Bước 9: Đối tượng Printer sẽ chỉ đến máy in mặc định. Trong hàm xử lý sự kiện Command4_Click chèn thêm đoạn mã:

Printer.Print Text1.Text

Câu lệnh này dùng để in nội dung trong TextBox ra máy in. Tuy nhiên nội dung của TextBox chỉ được in khi chương trình chấm dứt. Để in ngay lập tức, ta cần phải thêm dòng sau:

Printer.EndDoc

CHÉP DỮ LIỆU VÀO CLIPBOARD

Bước 10: Trong nhiều ứng dụng, nhiều khi ta cần sử dụng dữ liệu qua lại với nhau. Chẳng hạn, người dùng có thể sử dụng dữ liệu được hiển thị trên form hiển thị của chương trình chúng ta sang chương trình xử lý văn bản Microsoft Word. Lúc này, một cách hiệu quả nhất là sử dụng đối tượng Clipboard, đối tượng này cho phép đọc và ghi lên Windows Clipboard từ chương trình ứng dụng:

Thêm đoạn mã sau vào hàm sự kiện Command5_Click:

Clipboard.Clear

Clipboard.SetText Text1.Text

Đóng cửa số mã lệnh lại và chạy chương trình ứng dụng. Nhập một đoạn văn bản, sau đó nhấp Command5. Từ Microsoft Word, sử dụng menu Edit\Paste để lấy dữ liệu từ Clipboard hiển thị.

Bài tập 3-2

THAO TÁC VỚI RESOURCE FILE

Mục tiêu: Giúp làm quen với tập tin resource của VB, nhất là củng cố các thao tác trên tập tin.

Bước 1: Tạo thư mục Basic\Bt3-2. Tạo một dự án mới trong thư mục này.

Bước 2: Tạo giao diện như hình sau:

[missing_resource: .png]

51234679108

Hình III.2: Tập tin resource

Trong đó:

Item 1: Caption: Thong tin nguoi su dung

BorderStyle: 3-Fixed Dialog

StartUpPosition: 2-Center Screen

Item 2: Label

Name: lblHelp

Index: 0

Item 3: Label

Name: lblHelp

Index: 1

Item 4: Label

Name: lblHelp

Index: 2

Item 5: TextBox

Name: txtHelp

Index: 0

Item 6: TextBox

Name: txtHelp

Index: 1

Item 7: TextBox

Name: txtHelp

Index: 2

Item 8: CommandButton

Name: cmdHelp

Index: 0

Item 9: CommandButton

Name: cmdHelp

Index: 1

Item 10: CommandButton

Name: cmdHelp

Index: 2

Bước 3: Ta nhận thấy các điều khiển có cùng một tên hiển thị (Thong tin). Mục tiêu của ta là sử dụng tập tin resource (tài nguyên) để thay đổi tên hiển thị trên các điều khiển. Để tạo tập tin tài nguyên, ta vào mục ADD-IN\ADD-IN MANAGER trên menu của VB. Trong các mục của ADD-IN MANAGER nhấp đúp vào resource editor và đóng mục ADD-IN MANAGER lại.

Bước 4: Chọn Tools\Resource Editor trên menu. Mở String Table Editor bằng cách nhấp chuột lên biểu tượng abc của Resource Editor. Cửa sổ soạn thảo cho tập tin tài nguyên sẽ mở ra. Ta nhập các hàng như sau:

ID	RESOURCE STRING	ID	RESOURCE STRING
1	Ten	7	So dt
2	Ho	8	So CMND
3	Ma nv	9	T. trang hn
4	Huy bo	10	Huy bo
5	Vo hieu hoa	11	Ve truoc
6	Ke	12	Hoan tat

Bước 5: Lưu tập tin tài nguyên lại.

Bước 6: Mở cửa sổ soạn thảo mã lệnh. Tạo kiểu do người dùng định nghĩa để lưu dữ liệu cần nhập vào. Thêm đoạn mã sau:

Private Type yeucau

ho As String

ten As String

manv As String

sodt As String

socmnd As String

tinhtranghn As String

End Type

Bước 7: Ta thêm 2 biến nữa; một biến lưu thông tin về người sử dụng (theo kiểu ở trên), một biến lưu thứ tự các bước mà người sử dụng đã

nhập thông tin của mình vào.

Private chisobuoc As Integer

Private cacyeucau As yeucau

Bước 8: Trong chương trình này, người sử dụng phải nhập thông tin của mình vào thông qua các bước nhập, trong đó các điều khiển được sử dụng như một mảng các điều khiển. Để tận dụng chúng ta cần khai báo các hằng số để biết hiện thời người dùng đang ở bước thứ mấy của quá trình nhập thông tin cũng như biết được mình đã nhấp vào nút nhấn nào trong quá trình trên. Do đó, ta thêm đoạn khai báo sau:

' cac hang so

Private Enum buoc

buoc1 = 1

buoc2 = 2

buoc3 = 3

End Enum

Private Enum nhannut

nuttrai

nutgiua

nutphai

End Enum

Bước 9: Chương trình này thể hiện trên một form duy nhất và sử dụng mảng các điều khiển để tạo các bước để người dùng nhập thông tin vào. Do đó ta sử dụng tập tin tài nguyên để hiển thị các tên của điều khiển nhằm hiển thị cho chính xác. Vì thế ta cần có một hàm (thủ tục) để cập

nhật thông tin nhập vào dựa vào các bước của người dung khi nhập thông tin vào. Vào Tools\Add Procedure để thêm thủ tục sau:

```
Public Sub Hienthi()

Dim i As Integer

' Kiem tra cac buoc

Debug.Assert chisobuoc = 1 Or chisobuoc = 2

For i = 0 To 2

‘ Nhan

lblHelp(i).Caption = LoadResString((chisobuoc - 1) * 6 + (i + 1))

' Nut

cmdHelp(i).Caption = LoadResString((chisobuoc - 1) * 6 + (i + 4))

If UCase(cmdHelp(i).Caption) = "VO HIEU HOA" Then

cmdHelp(i).Visible = False

Else

cmdHelp(i).Visible = True

End If

txtHelp(i).Text = ""

Next

End Sub
```

Bước 10: Khi chương trình thực hiện, ta phải ở bước thứ nhất của quá trình nhập liệu ⇒ Thêm đoạn mã sau trong thủ tục xử lý sự kiện

Form_Load:

chisobuoc = 1

Hienthi

Bước 11: Mỗi khi có một nút nhấn được nhấp, quá trình nhập liệu chuyển sang bước kế tiếp; người sử dụng có thể đi đến bước kế tiếp hay trở về bước trước đó trong quá trình này. Vì các nút nhấn (button) là một mảng điều khiển (control array) nên chúng có cùng một sự kiện Click tác động vào gọi là cmdHelp_Click. Hàm xử lý này có tham số là một chỉ số kiểu Integer để nhận biết nút nhấn nào được nhấp. Ở đây, ta thêm đoạn mã sau trong hàm xử lý sự kiện này.

Private Sub cmdHelp_Click(Index As Integer)

Select Case chisobuoc

Case buoc1

cacyeucau.ten = txtHelp(0).Text

cacyeucau.ho = txtHelp(1).Text

cacyeucau.manv = txtHelp(2).Text

Case buoc2

cacyeucau.sodt = txtHelp(0).Text

cacyeucau.socmnd = txtHelp(1).Text

cacyeucau.tinhtranghn = txtHelp(2).Text

End Select

' Cac nut nhan

Select Case Index

Case nuttrai

' Huy bo

End

Case nutgiua

' ve truoc

chisobuoc = buoc1

Hienthi

Case nutphai

' di toi

chisobuoc = chisobuoc + 1

If chisobuoc = buoc2 Then

Hienthi

Else

Guiyeucau

End If

End Select

End Sub

Bước 12: Khi quá trình nhập thông tin kết thúc, thông tin này được lưu vào trong một tập tin văn bản, nhờ thủ tục Guiyeucau. Thêm thủ tục Guiyeucau vào nhờ mục Tools\Add Procedure và nhập đoạn mã sau:

Public Sub Guiyeucau()

```
On Error GoTo Guiloi
' Lay the tap tin
Dim intFile As Integer
intFile = FreeFile()
' Viet len tap tin
Open App.Path & "\yeucau.txt" For Output As #intFile
Print #1, "ho: " & cacyeucau.ho
Print #1, "ten: " & cacyeucau.ten
Print #1, "manv: " & cacyeucau.manv
Print #1, "sodt: " & cacyeucau.sodt
Print #1, "socmnd: " & cacyeucau.socmnd
Print #1, "tinhtranghn: " & cacyeucau.tinhtranghn
Close #intFile
MsgBox "Yeu cau cua ban da duoc goi di", vbOKOnly + vbInformation, _
"Goi yeu cau"
End
Exit Sub
Guiloi:
MsgBox Err.Description, vbOKOnly + vbExclamation, "Goi yeu cau"
Exit Sub
```

End Sub

Bước 13: Lưu và thực thi chương trình.

Bài tập 3-3

CHƯƠNG TRÌNH XỬ LÝ VĂN BẢN ĐƠN GIẢN

GIAO DIỆN ĐA TÀI LIỆU

Bước 1: Tạo một dự án lưu trong thư mục Basic\Bt3-3.

Giao diện đa tài liệu (MDI Form) gồm một cửa sổ cha chứa nhiều cửa sổ con (chẳng hạn như các chương trình Microsoft Word, Excel được tổ chức theo dạng này). Để thêm vào dự án, ta chọn mục Project\Add MDI Form từ menu của VB.

Bước 2: Ta cho Form1 trở thành một cửa sổ con của MDI Form bằng cách chọn thuộc tính MDIChild = True.

HÀM MAIN (SUB MAIN)

Bước 3: Trong chương trình ta cần điều khiển mọi thứ kể từ khi các cửa sổ con của MDI Form xuất hiện, do đó ta cần phải bắt đầu thực thi chương trình của ta từ hàm Main (Sub Main). Ta chọn mục Project\ Add Module để thêm một Modul vào dự án của mình, sau đó ta chọn Tools\Add Procedure để thêm hàm Main vào (Public Sub Main); hàm này ta dùng để bắt đầu gọi thực thi chương trình của mình. Để chọn thực thi chương trình từ hàm Main, chọn Project\Properties; chọn Start up Object là Sub Main.

Bước 4: Thêm dòng lệnh sau vào hàm Main:

MDIForm1.Show

Bước 5: Chương trình cần có một hệ thống menu để gọi thực thi. Do đó, chọn MDI Form, sau đó chọn Tools\Menu Editor để tạo menu sau:

Menu NameMenu Caption

mnuFile&File

mnuFileNew &New

mnuFileOpen&Open…

muFileSave&Save

mnuFileBar-

mnuFileExitE&xit

Bước 6: Ta xử lý sự kiện mnuFileExit_Click nhờ đoạn mã sau:

```
Private Sub mnuFileExit_Click()
Dim f As Form
' Thoat cac cua so con
For Each f In Forms
If TypeOf f Is Form1 Then
Unload f
Set f = Nothing
End If
Next
' Thoat cua so cha
Unload Me
End Sub
```

Bước 7: Để tạo ra một tài liệu trắng cho chương trình xử lý văn bản, ta cần phải có một TextBox trong Form1. Người sử dụng đánh nội dung vào TextBox, do đó ta thêm một TextBox vào Form1 với các thuộc tính sau:

MultiLine: True

ScrollBars: 2-Vertical

Ta xử lý sự kiện Form_Resize của Form1 như sau:

```
Private Sub Form_Resize()

Text1.Height = Me.ScaleHeight

Text1.Width = Me.ScaleWidth

Text1.Left = 0

Text1.Top = 0

End Sub
```

Bước 8: Mỗi lần chọn mục New trên cửa sổ chương trình ứng dụng, một khung cửa sổ trắng hiện ra để ta nhập văn bản vào. Do đó, thêm đoạn mã sau trong thủ tục xử lý sự kiện mnuFileNew_Click:

```
Private Sub mnuFileNew_Click()

Dim f As Form1

Static n As Integer

Set f = New Form1

f.Text1.Text = ""

n = n + 1

f.Caption = "Document " & Format(n)
```

```
f.Show

End Sub
```

THAO TÁC TRÊN TẬP TIN

Bước 9: Ta cần phải có hộp thoại nhằm chọn tập tin để lưu (hay mở tập tin) trong chương trình xử lý văn bản. Do đó ta cần thêm một Dialog Control vào chương trình. Đánh dấu vào mục chọn Microsoft Common Dialog Control 6.0 (SP3). Sau đó ta thêm Dialog Control từ ToolBox vào MDIForm1. Ta xử lý sự kiện mnuFileSave_Click nhờ đoạn mã sau:

```
Private Sub menuFileSave_Click()

Dim tenfile As String

CommonDialog1.ShowSave

tenfile = CommonDialog1.FileName

Open tenfile For Output As #1

Print #1, MDIForm1.ActiveForm.Text1.Text

Close #1

End Sub
```

Bước 10: Khi mục Open của menu được chọn, hộp thoại Open File được mở ra $\Rightarrow$ sự kiện mnuFileOpen_Click được xử lý như sau:

```
Private Sub mnuFileOpen_Click()

Dim tenfile As String, s As String

CommonDialog1.ShowOpen

tenfile = CommonDialog1.FileName
```

```
If UCase(Right(tenfile, 3)) <> "TXT" Then Exit Sub

Call mnuFileNew_Click

Open tenfile For Input As #1

Do Until EOF(1)

Line Input #1, s

Me.ActiveForm.Text1.Text = Me.ActiveForm.Text1.Text & s & vbCrLf

Loop

Close #1

End Sub
```

Bước 11: Lưu dự án và chạy chương trình. Tạo mới, lưu, mở một số tài liệu. Nhận xét kết quả.

Bài tập 3-4

THAO TÁC VỚI ĐỐI TƯỢNG WORD

Mục đích: Windows có sẵn một số đối tượng khi ta cài đặt Windows hay khi cài một số phần mềm. Bài tập này giúp ta tìm hiểu cách thức truy xuất các đối tượng có sẵn này từ Visual Basic.

THAM CHIẾU ĐỐI TƯỢNG

Bước 1: Tạo thư mục Basic\Bt3-4. Khởi động một dự án mới trong thư mục này.

Bước 2: Trong bài tập này ta có tham chiếu đến đối tượng Word của Microsoft Word; do đó ta phải có thao tác tham chiếu đến đối tượng này trong màn hình soạn thảo VB bằng cách: Chọn Project\References trên

menu. Trong cửa sổ References, thiết lập tham chiếu đến: Microsoft Word 9.0 Object Library và Microsoft Office 9.0 Library. Sau đó đóng cửa sổ References lại.

Bước 3: Ta có thể kiểm tra các đối tượng trên có được đưa vào hay chưa nhờ thao tác: Chọn View\Object Browser.

XÂY DỰNG ỨNG DỤNG

Bước 4: Tạo giao diện chương trình có dạng sau:

[missing_resource: .png]

123456

Hình III.3: Thao tác với đối tượng Word

Trong đó:

1: TextBox

Name: txtWord

Multiline: True

ScrollBar: 2-Vertical

2: CommandButton

Name: cmdLuu

Caption: Lưu

3: CommandButton

Name: cmdTruoc

Caption: Trước khi in

4: CommandButton

Name: cmdCTa

Caption: Kiểm lỗi.

5: CommandButton

Name: cmdThoat

Caption: Thoát

6: CommandButton

Name: cmdGiup

Caption: Trợ giúp

Bước 5: Để sử dụng được mô hình, ta phải khai báo một số biến đối tượng của Word. Trong phần [General]\ [Declarations], khai báo những biến sau:

Public ungdung As Word.Application

Public tailieu As Word.Document

Public trogiup As Office.Assistant

Bước 6: Khi chương trình thực hiện, điều ta muốn là một tài liệu mới của Word được tạo ra để ta có thể thao tác trên chúng một cách gián tiếp thông qua chương trình VB của mình. Tạo một tài liệu Word mới tương đương với việc tạo ra một thể hiện của đối tượng Document. Vì thế, chèn đoạn mã sau vào thủ tục Form_Load để tạo ra một tài liệu Word mới từ chương trình VB.

Set ungdung = CreateObject("Word.Application")

Set tailieu = ungdung.Documents.Add

Set trogiup = ungdung.Assistant

Bước 7: Nút Lưu có nhiệm vụ ghi tất cả những gì trên TextBox vào đối tượng Word mới tạo ra. Do đó, ta xử lý sự kiện cmdLuu_Click như sau:

' Ghi tai lieu moi

tailieu.Content.Text = txtWord.Text

MsgBox "Van ban duoc luu trong Word", vbOKOnly, "Word"

Bước 8: Nút Trước khi in có nhiệm vụ hiển thị tài liệu Word giống như khi chúng được in ra giấy; vì thế sự kiện cmdTruoc_Click được xử lý như sau:

tailieu.PrintPreview

ungdung.Visible = True

ungdung.Activate

Bước 9: Nút Kiểm lỗi thực hiện thao tác kiểm lỗi chính tả cho tài liệu Word, thao tác này được xử lý trong thủ tục cmdCTa_Click:

tailieu.CheckSpelling

txtWord.Text = tailieu.Content.Text

Bước 10: Nút Thoát sẽ đóng cửa sổ chứa tài liệu Word lại. Chèn đoạn mã sau trong thủ tục cmdThoat_Click để đóng Word lại:

ungdung.Quit SaveChanges = False

Bước 11: Khi ta nhấp vào nút Trợ giúp thì cửa sổ Help của Office hiện ra. Do đó, thêm đoạn mã sau trong thủ tục cmdGiup_Click để mở cửa sổ

Help của Office:

ungdung.Visible = True

ungdung.Activate

trogiup.Help

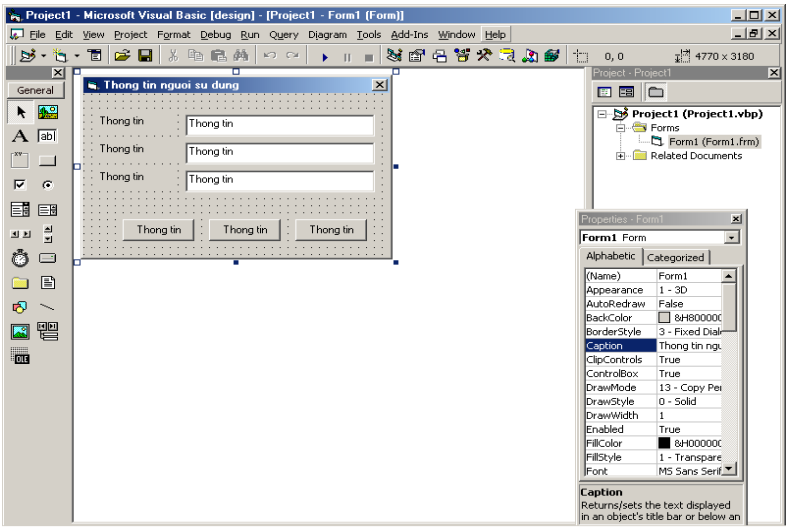
Bước 12: Lưu dự án và chạy chương trình. Nhập vài dòng văn bản vào TextBox và nhấp nút Lưu. Mỗi khi văn bản được lưu, ta nhấp nút Trước khi in để xem tài liệu trước khi. Khi văn bản được hiển thị, nhấp nút Kiểm lỗi. Sau đó thử nhấp nút Trợ giúp để xem phần giúp đỡ của Office. Cuối cùng nhấp nút Thoát để thoát khỏi Word.

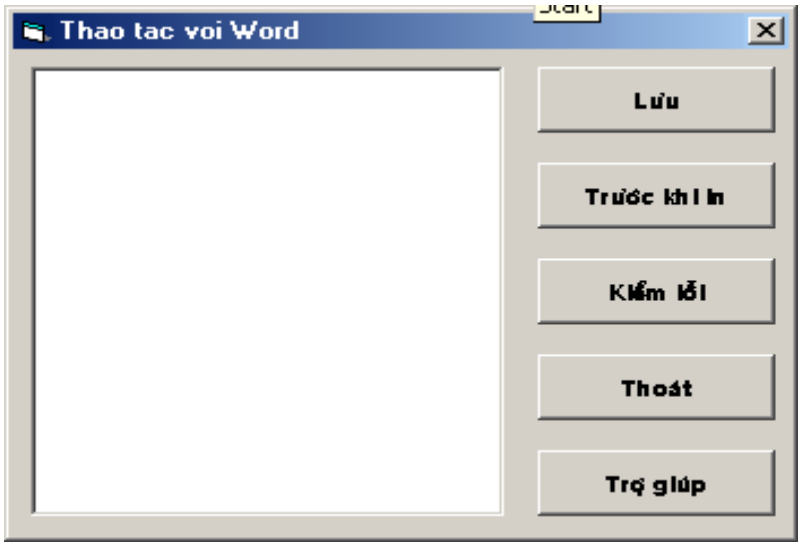
BÀI TẬP TỰ LÀM

1. Thiết kế chương trình như ứng dụng Notepad của Windows.

- Thiết kế giao diện

Hình III.5: Giao diện Notepad





- Xử lý các sự kiện

- Mỗi khi Form thay đổi kích thước, TextBox cũng thay đổi theo cho phù hợp với Form
- New
- Open, Save, Save As: mở hộp thoại Common Dialog cho phép chọn tập tin để mở hay lưu. Sử dụng đối tượng FileSystemObject để thao tác với tập tin văn bản.
- Exit
- Font Setting: Mở ra hộp thoại chọn Font, thiết lập Font của TextBox chính là Font được chọn trong hộp thoại.

- Xử lý mở rộng:

- Khi người dùng đã lưu tập tin rồi, lần thứ hai bấm vào Save thì không mở hộp thoại Common Dialog nữa mà sẽ lưu với tên tập tin đã chọn trong lần Save đầu tiên.
- Mỗi khi người dùng thay đổi nội dung của một tập tin, sau đó họ chọn Exit để đóng ứng dụng lại; một hộp thông điệp (Message Box) sẽ mở ra hỏi có lưu tập tin hay không?
- Đại lý Minh Thành của công ty Unilever Việt Nam tại Cần Thơ cần quản lý thông tin về các mặt hàng mà đại lý nhận từ công ty. Các thông tin cần quản lý gồm: Mã mặt hàng, tên mặt hàng, đơn vị tính, giá của mặt hàng đó. Các thông tin này được mô tả như sau:

Type HangHoa

MaHang As String*5

TenHang As String*40

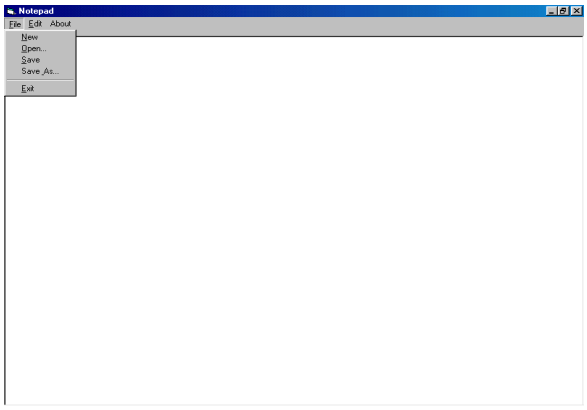
DVTinh As String*15

Gia As Double

End Type

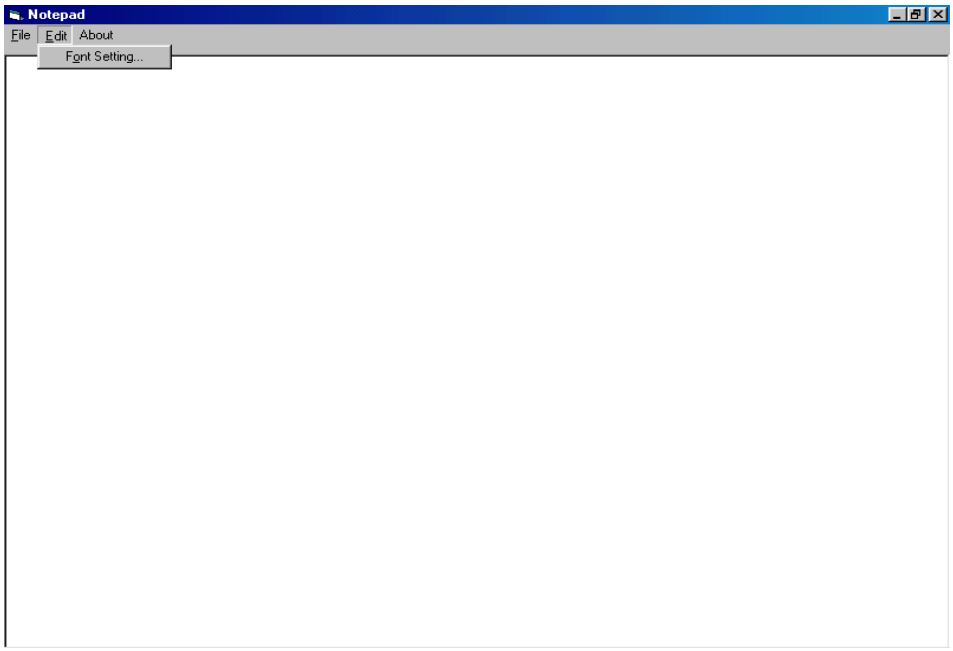
Dựa vào thông tin mô tả trên, Anh (Chị) hãy:

1. Tạo dự án mới và viết các khai báo thích hợp.
2. Thiết kế Form chính như sau:



Hình III.6: Giao diện chính

1. Khi người dùng chọn mục Thoát, rồi nhấp chọn Thực hiện, chương trình chấm dứt. Viết mã lệnh để xử lý đối với trường hợp này.
2. Khi người dùng nhấp chọn Nhập liệu, rồi Thực hiện, một Form sẽ mở ra cho phép nhập thông tin hàng hóa vào. Hãy thiết kế Form này với các chức năng như hình dưới:



Hình III.7: Form nhập liệu

1. Mỗi khi người dùng nhập thông tin vào các ô TextBox, rồi chọn nút nhấn Nhập, những thông tin đó sẽ được lưu lên lưới hiển thị. Khi chọn Ghi tập tin, một hộp thoại (Common Dialog) lưu tập tin hiện ra cho phép chọn đường dẫn và tên tập tin, sau đó ghi những thông tin trên lưới vào tập tin đã chọn (với cấu trúc tập tin được mô tả ở phần đầu). Nút nhấn Thoát sẽ đóng Form này lại, trở về Form chính ban đầu. Viết các đoạn xử lý thích hợp.
2. Ở Form chính ban đầu (hình III.6), khi người dùng chọn Hiển thị thông tin hàng hóa, một hộp thoại mở tập tin (CommonDialog) hiện ra cho phép chọn tên tập tin chứa dữ liệu về hàng hóa đã được tạo ra ở câu e. Sau đó đọc dữ liệu từ tập tin rồi hiển thi trên lưới:

Hình III.8: Đọc từ tập tin

Hãy thiết kế Form và viết mã lệnh xử lý các sự kiện thích hợp.

CƠ SỞ DỮ LIỆU SỬ DỤNG

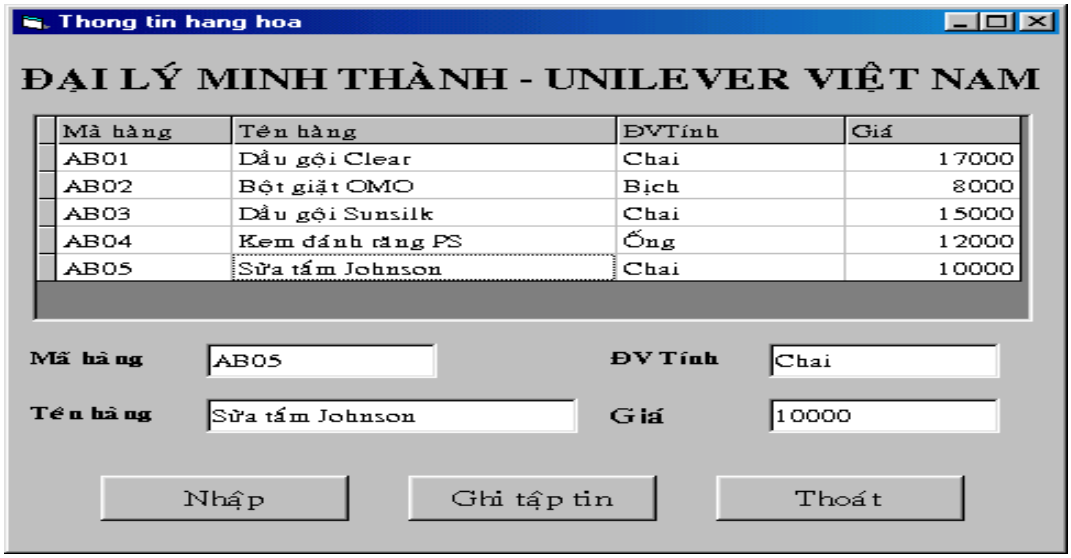
Các chương kế tiếp là phần lập trình Visual Basic truy xuất cơ sở dữ liệu (CSDL). Trong các bài tập trên CSDL, ta có sử dụng CSDL HangHoa.MDB của Access. Cơ sở dữ liệu này đã có sẵn, sinh viên có thể liên hệ cán bộ giảng dạy để lấy về. Thông tin các bảng (Table) của CSDL này như sau:

TLOAIHANG(MaLoai, TenLoai): Mỗi loại hàng hóa có mã loại và tên loại.

THANGHOA(MaHang, TenHang, DVTinh, MaLoai). Mỗi hàng hóa có mã hàng hóa, tên hàng hóa, đơn vị tính và chỉ thuộc 1 loại hàng hóa nào đó.

TNHANVIEN(MaNV, HoTen, Phai, Diachi, Ngaysinh, Luong, Ghichu): Mỗi nhân viên có mã nhân viên, họ tên, phái, địa chỉ nhân viên, ngày sinh, lương của nhân viên đó là bao nhiêu và có thể có một vài ghi chú về nhân viên đó.

TPHATSINH(SOTT, Ngay, Loai, Fieu, Hten, Lydo, MaHang, Solg, Dgia, MaNV): Mỗi một phát sinh được ghi nhận thành một chứng từ có SoTT, ngày phát sinh chứng từ, loại phát sinh là nhập (hay xuất)…, số phiếu, họ tên khách hàng, lý do phát sinh ứng với hàng hóa nào (mã hàng), số lượng và đơn giá là bao nhiêu, nhân viên phụ trách phát sinh là gì (MaNV).



Bảng quan hệ giữa các Table này như sau:

Các mối quan hệ của CSDL HangHoa.mdb

Khái niệm cơ bản về cơ sở dữ liệu
Mục tiêu: Chương này giới thiệu về một số khái niệm trong lập trình cơ sở dữ liệu với VB, những vấn đề cần thiết khi thiết kế các ứng dụng truy cập cơ sở dữ liệu.

Học xong chương này, sinh viên phải nắm bắt được các vấn đề sau:

- Một số khái niệm khi lập trình cơ sở dữ liệu trong VB.

- Phân biệt DAO, RDO, ADO.

- Sử dụng môi trường phát triển VB để tương tác với cơ sở dữ liệu.

Kiến thức có liên quan:

- Câu lệnh SQL để truy vấn dữ liệu.

Tài liệu tham khảo:

- Microsoft Visual Basic 6.0 và Lập trình Cơ sở dữ liệu - Chương 18, trang 447 - Nguyễn Thị Ngọc Mai (chủ biên), Nhà xuất bản Giáo dục – 2000.

Cơ sở dữ liệu

Khái niệm

Cơ sở dữ liệu là một kho chứa thông tin. Có nhiều loại cơ sở dữ liệu, nhưng trong khuôn khổ bài giảng này ta chỉ quan tâm đến các ứng dụng lập trình liên quan đến cơ sở dữ liệu quan hệ.

Một cơ sở dữ liệu quan hệ:

- Chứa dữ liệu trong các bảng, được cấu tạo bởi các dòng còn gọi là các mẩu tin, và cột còn gọi là các trường.
- Cho phép lấy về (hay truy vấn) các tập hợp dữ liệu con từ các bảng.

- Cho phép nối các bảng với nhau cho mục đích truy cập các mẩu tin liên quan với nhau chứa trong các bảng khác nhau.

Bộ máy (Engine) cơ sở dữ liệu

Chức năng cơ bản của một cơ sở dữ liệu được cung cấp bởi một bộ máy cơ sở dữ liệu, là hệ thống chương trình quản lý cách thức chứa và trả về dữ liệu.

Chẳng hạn Microsoft Jet là bộ máy cơ sở dữ liệu được sử dụng khi truy cập dữ liệu Access.

Bảng (Table) và trường (Field)

Các cơ sở dữ liệu được cấu thành từ các bảng dùng thể hiện các phân nhóm dữ liệu. Chẳng hạn, nếu ta tạo một cơ sở dữ liệu để quản lý các tài khoản trong công việc kinh doanh, ta phải tạo một bảng cho Khách hàng, một bảng cho Hóa đơn và một bảng cho Nhân viên. Bảng có cấu trúc định nghĩa sẵn và chứa dữ liệu phù hợp với cấu trúc này.

- Bảng: chứa các mẩu tin là các mẩu dữ liệu riêng rẽ bên trong phân nhóm dữ liệu.
- Mẩu tin: chứa các trường. Mỗi trường thể hiện một bộ phận dữ liệu trong một mẩu tin. Ví dụ như mỗi mẩu tin thể hiện một mục trong danh bạ địa chỉ chứa các trường tên và họ, địa chỉ, thành phố, số điện thoại…

Ta có thể dùng chương trình Visual Basic để tham chiếu và thao tác với cơ sở dữ liệu, bảng, mẩu tin và các trường.

Tập mẩu tin (Recordset)

Recordset là một cấu trúc dữ liệu thể hiện một tập hợp con các mẩu tin lấy về từ cơ sở dữ liệu. Về khái niệm, nó tương tự như một bảng nhưng có thêm một vài thuộc tính riêng biệt quan trọng.

Các Recordset được thể hiện như các đối tượng. Cũng như các đối tượng khác trong Visual Basic, các đối tượng recordset có các thuộc tính và phương thức riêng.

Truy xuất cơ sở dữ liệu trong Visual Basic 6.0

Visual Basic cung cấp kèm theo nó một bộ máy cơ sở dữ liệu có thể hiểu được dữ liệu của Microsoft Access gọi là Joint Engine Technology (JET). JET là một bộ máy truy cập cơ sở dữ liệu hướng đối tượng và nó là một phần không thể thiếu được của Visual Basic. Phiên bản của JET đi kèm với VB 6.0 là miễn phí nghĩa là VB có thể truy xuất trực tiếp cơ sở dữ liệu của Microsoft Access. Giao diện để VB truy xuất JET có tên là Data Access Objects (DAO).

JET là một bộ máy cơ sở dữ liệu tuyệt vời cho các ứng dụng văn phòng chạy trên máy đơn, nhưng hiệu suất của nó giảm đáng kể khi số lượng người dùng tăng lên và cơ sở dữ liệu được mở rộng. Vì điều này JET không phải là một giải pháp tối ưu cho các ứng dụng cơ sở dữ liệu nhiều người dùng. Cho đến nay người ta chưa có một thống kê chính xác được kích thước dữ liệu tối đa hay số lượng người dùng tối đa của JET nhưng nhìn chung JET bị giới hạn nhiều hơn so với các giải pháp khác trong môi trường đa người dùng. Tuy vậy, JET là điểm khởi đầu tốt nhất cho người lập trình VB bởi vì sự đơn giản của nó.

Khi kích thước dữ liệu tăng lên, người lập trình bao giờ cũng muốn xây dựng một ứng dụng Khách/Chủ (Client/Server) có khả năng bảo mật cao và linh hoạt. Vì lẽ đó, Microsoft hỗ trợ trong VB để truy cập các cơ sở dữ liệu quan hệ được thông dịch bởi chuẩn Open Database Connectivity (ODBC). ODBC là một kỹ thuật cho phép truy cập các cơ sở dữ liệu quan hệ cao cấp như SQL SERVER hay ORACLE. Tuy nhiên, ODBC cũng có thể được sử dụng để truy cập các cơ sở dữ liệu nhỏ tổ chức bằng Microsoft Access hay Foxpro, thậm chí các cơ sở dữ liệu máy chủ như IBM DB2. Visual Basic sử dụng giao diện đối tượng Remote Data Objects (RDO) để truy cập ODBC.

DAO và RDO là những kỹ thuật hỗ trợ việc truy xuất đến các cơ sở dữ liệu quan hệ. Tuy nhiên, Microsoft lại cung cấp một công cụ hữu ích hơn để truy cập dữ liệu goi là OLEDB. OLEDB là kỹ thuật cho phép dữ liệu được truy xuất từ cả 2 nguồn cơ sở dữ liệu: quan hệ và không quan hệ. Điều đó có nghĩa là gồm các cơ sở dữ liệu của Microsoft Access, Oracle, SQL SERVER và cả các nguồn dữ liệu không quan hệ như Excel, Microsoft Index Server, Microsoft Exchange, Active Directory… Visual Basic sử dụng giao diện đối tượng ActiveX Data Objects (ADO) để truy cập OLEDB.

Visual Basic cung cấp cho ta nhiều công cụ để truy cập dữ liệu như DAO, RDO, ADO. Câu hỏi thường đặt ra là: Kỹ thuật nào được sử dụng lúc nào ở đâu? Nhiều người cho rằng DAO & RDO đã lỗi thời và người ta hiếm sử dụng chúng. Thật ra DAO & RDO là các điển hình cho một vài khả năng tiêu biểu của ADO. Hiện nay, vẫn còn khá nhiều ứng dụng sử dụng DAO & RDO và thật sự chúng bị giới hạn trong chừng mực nào đó. OLEDB thực sự cung cấp một khả năng rộng lớn để truy cập các cơ sở dữ liệu từ nhiều nguồn khác nhau. Tuy vậy, trong một số trường hợp một giải pháp dùng RDO lại hữu dụng hơn ADO.

Dùng Visual Basic để tạo một cơ sở dữ liệu

Thông thường chúng ta sẽ sử dụng các hệ quản trị cơ sở dữ liệu để tạo nên một cơ sở dữ liệu, nhưng trong phần này ta sẽ xét qua tính năng tạo cơ sở dữ liệu bằng Visual Basic 6.0. Ta có thể áp dụng phương pháp này cho những cơ sở dữ liệu nhỏ và tương thích với Microsoft Access.

Sử dụng cửa sổ cơ sở dữ liệu

- Từ Menu của VB6, chọn mục Add-Ins, Visual Data Manager. Cửa sổ Visual Data Manager sẽ xuất hiện.
- Chọn mục File -> New -> MicroSoft Access -> Version 7.0 MDB.
- Chọn thư mục ta muốn lưu cơ sở dữ liệu và tên của cơ sở dữ liệu.

[missing_resource: .png]

Hình VIII.1 Cửa sổ Visual Data Manager

Tạo bảng

- Để tạo mới một bảng, ta chọn Properties trong cửa sổ Databases, nhấp chuột phải, chọn New Table, đặt tên cho Table tại ô Table Name, ấn Add Field để tạo mới các trường cho bảng.

[missing_resource: .png]

Hình VIII.2 Cửa sổ tạo Table

- Ta sẽ nhập tên trường tại ô Name, chọn kiểu của trường tại Combo Type, tùy chọn FixedField và VariableField xác định độ dài của trường là cố định hay thay đổi.
- Sau khi xác định đầy đủ các thuộc tính của trường, ấn OK và tiếp tục thêm vào các trường khác cho bảng. Nếu đã thêm mới đầy đủ các trường của bảng, ấn Close để quay về cửa sổ Table Structure.
- Sau khi quay về cửa sổ Table Structure, ta sẽ xác lập các chỉ mục cũng như khóa chính của bảng.

[missing_resource: .png]

Hình VIII.3 Cửa sổ tạo khóa chính và chỉ mục

- Tại ô Name, ta sẽ nhập vào tên của chỉ mục, rồi chọn các trường tham gia vào chỉ mục đó. Nếu ta chọn Primary thì đó chính là các trường cấu thành khóa chính của bảng. Chọn Unique tức là giá trị của chỉ mục đó sẽ không có sự trùng lặp.
- Ấn Close xác nhận rằng ta đã xây dựng xong tập các chỉ mục của bảng.

- Sau khi đã hoàn thành tất cả các thao tác trên, để tạo bảng ta ấn Build the Table.

Tuy rằng đây là một tính năng mới của VB6, tuy nhiên chúng ta cũng sẽ gặp phải rất nhiều bất tiện khi phải thiết kế một cơ sở dữ liệu hoàn chỉnh cũng như trong quá trình bảo trì và sử dụng (khó khăn trong việc thay đổi các thuộc tính đã xác lập, không tạo liên kết giữa các bảng được …). Một phương cách tốt nhất đó là nên dùng các hệ quản trị cơ sở dữ liệu chuyên dùng để thực hiện công việc nêu trên.

[missing_resource: .png]

Hình VIII.4 Tạo bảng cho cơ sở dữ liệu

Dùng Visual Data Manager để tạo giao diện

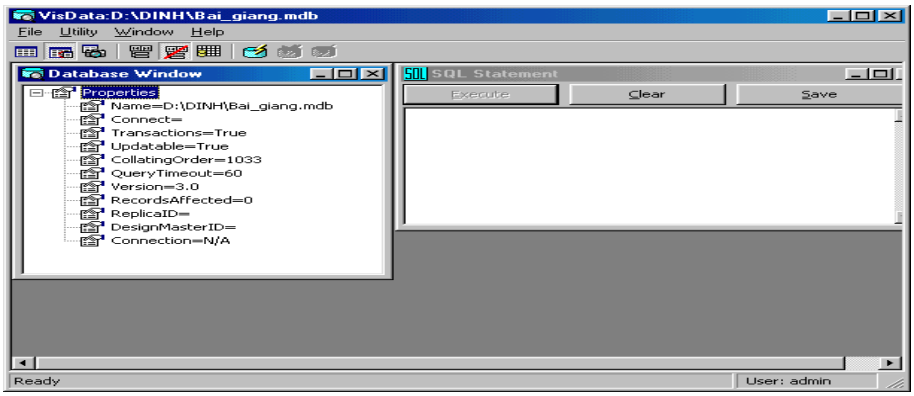
Ta có thể thiết kế một Form nhập liệu đơn giản cho một Table từ Visual Data Manager. Các bước tiến hành như sau:

- Từ Visual Data Manager chọn Cơ sở dữ liệu cần thao tác.
- Chọn Data Form Design từ mục Utility.
- Chọn Table cần cho việc tạo Form và các trường hiển thị trên Form (thông thường chúng ta sẽ cho hiển thị tất cả các trường).
- Chọn Build the Form, biểu mẫu mới đã được tạo trong đề án của chúng ta.

[missing_resource: .png]

Hình VIII.5 Thiết lập các thuộc tính cho Form

Kết quả sau khi chúng ta xây dựng Form bằng Visual Data Manager:



Sử dụng cửa sổ xem dữ liệu (Data View)

VB6 còn hỗ trợ cho người lập trình khả năng làm việc với cơ sở dữ liệu mà không phải thông qua công cụ quản trị cơ sở dữ liệu hoặc các công cụ trong Add-In. Công cụ này chính là cửa sổ Data View.

- Từ Menu của VB chọn Data View hoặc nhấn nút Data View trên thanh công cụ.
- Cửa sổ Data View xuất hiện cho ta hai lựa chọn: Data Links và Data Environment Connections.

[missing_resource: .png]

Hình VIII.6 Cửa sổ Data View

Liên kết dữ liệu (Data Link) là một cách kết nối môi trường phát triển của VB6 với một cơ sở dữ liệu nào đó. Một kết nối môi trường dữ liệu (Data Environment Connection) là cách thức sử dụng cơ sở dữ liệu trong một đề án cụ thể. Điểm khác biệt giữa hai thành phần này là sự liên quan giữa cơ sở dữ liệu với đề án cụ thể.

Để sử dụng cơ sở dữ liệu theo kiểu Data Link, ta tiến hành như sau:

- Chọn Data Links, ấn chuột phải -> Add a Data Link hoặc ấn nút Data Link trên thanh công cụ của cửa sổ.
- Cửa sổ Data Link Properties xuất hiện.

[missing_resource: .png]

Hình VIII.7 Hộp thoại Data Link Properties

- Chọn trình cung cấp Microsoft Jet, chọn Next.
- Chọn cơ sở dữ liệu muốn nối kết đến, nhấn OK.

Liên kết dữ liệu cung cấp một cách nhìn tóm lược về nguồn dữ liệu. Mỗi lần ta tạo một liên kết dữ liệu, ta có thể duyệt bằng cách mở rộng phần tử trong danh sách tóm lược. Thực hiện điều này bằng cách nhấn vào dấu cộng bên trái mỗi phần tử (hình VIII.8).

[missing_resource: .png]

Hình VIII.8 Cửa số Data View

Sử dụng điều khiển dữ liệu để tạo giao diện người sử dụng

Điều khiển dữ liệu giúp cho người sử dụng liên kết biểu mẫu của mình đến nguồn cơ sở dữ liệu. Điều khiển dữ liệu cung cấp cho người sử dụng những tính năng xử lý dữ liệu cơ bản như duyệt qua các mẩu tin, thêm mới, cập nhật.

Đối với phiên bản VB6 cung cấp cho chúng ta 3 trình điều khiển dữ liệu: DAO (Data Access Object), RDO (Remote Data Object) và ADO (ActiveX Data Object).

Kết nối với cơ sở dữ liệu và làm việc với các mẩu tin thông qua điều khiển ADO Data

Hiển thị dữ liệu

Nếu như chúng ta xây dựng một biểu mẫu chỉ để hiển thị các mẩu tin của một bảng, điều này rất đơn giản và ta không cần phải lập trình gì cả.

Để sử dụng điều khiển ADO Data, ta cần đánh dấu Microsoft ADO Data Control 6.0 (OLEDB) trong hộp thoại Components.

[missing_resource: .png]

Hình VIII.9 Hộp thoại Components

Chọn điều khiển ADO Data từ hộp công cụ đưa vào biểu mẫu, liên kết đến nguồn dữ liệu thông qua hai thuộc tính ConnectionString và RecordSource.

- ConnectionString: Xác định nguồn dữ liệu cần nối kết, đó chính là chuỗi nối kết chỉ đến cơ sở dữ liệu mà ta thao tác.
- RecordSource: Xác định xem nối kết của ta đang thao tác trên bảng nào.

Ví dụ:Tạo một nối kết đến cơ sở dữ liệu "C:\Program Files\Microsoft Visual Studio\VB98\Biblio.mdb".

- Chọn Use Connection String, ấn Build.
- Chọn Microsoft Jet 4.0 OLE DB Provider.
- Chọn cơ sở dữ liệu như ví dụ.
- Ấn OK.
- Quay về cửa sổ Property Pages, chọn Tab RecordSource, xác định các tùy chọn như hình vẽ.
- Ấn Close.

[missing_resource: .png]

Hình VIII.10 Tùy chọn RecordSource

Sau khi đã xác định được nối kết, ta vẫn không thấy được sự hoạt động của điều khiển dữ liệu, nguyên nhân do chúng ta không có điều khiển để hiển thị nội dung, cách giải quyết vấn đề là dùng điều khiển TextBox hiển thị dữ liệu.

Để dùng điều khiển Textbox hiển thị dữ liệu, ta xác định hai thuộc tính sau đây của điều khiển: DataSource, DataField. Các thuộc tính này xác định nguồn dữ liệu và tên trường, đối với ví dụ này đó là Adodc1 (tên của ADO Data) và Au_Id.

- Thực thi đề án, ta được kết quả sau:

[missing_resource: .png]

Hình VIII.11 Ví dụ dùng ADO Data

Cập nhật dữ liệu

Thao tác cập nhật dữ liệu cũng khá đơn giản, điều khiển ADO Data sẽ tự động cập nhật lại giá trị của mẩu tin hiện hành mỗi khi ta duyệt qua mẩu tin khác, vì vậy ta cũng không phải làm gì cả.

Thêm mới mẩu tin

Để có thể thêm mới mẩu tin, ta có hai phương cách như sau:

- Thiết lập thuộc tính EOFAction của điều khiển ADO Data là 2-AddNew. Cách này không cần phải lập trình gì cả.

[missing_resource: .png]

Hình VIII.12 Thêm mới mẩu tin dùng ADO DataĐể thêm mới vào một mẩu tin, ta sẽ đi đến cuối mẩu tin, sau đó ấn nút tiếp, ta nhận thấy giá trị của các trường sẽ rỗng để chờ chúng ta nhập mới thông tin vào.

- Thêm mới mẩu tin bằng 2 phương thức AddNew và Update, các bước tiến hành sẽ như sau:

- Thiết kế hai nút lệnh là ADD NEW và UPDATE.

- Trong sự kiện Click của hai nút trên lần lượt nhập vào câu lệnh sau: Adodc1.Recordset.AddNew, Adodc1.Recordset.Update với Adodc1 là thuộc tính Name của điều khiển dữ liệu.

[missing_resource: .png]

Hình VIII.13 Sử dụng phương thức AddNew và Update

Dùng sự kiện MoveComplete để cập nhật giao diện người sử dụng

Ta có thể dùng sự kiện MoveComplete của điều khiển ADO Data để khởi động sửa đổi trong ứng dụng khi người sử dụng di chuyển từ mẩu tin này sang mẩu tin khác.

Sự kiện MoveComplete được kích hoạt khi một mẩu tin mới thành mẩu tin hiện hành. Đây là một trong vài sự kiện được kích hoạt khi điều khiển di chuyển từ mẩu tin này sang mẩu tin khác. Các sự kiện khác bao gồm WillChange, được kích hoạt khi điều khiển di chuyển từ mẩu tin này sang mẩu tin khác hay thay đổi một mẩu tin và sự kiện RecordChangeComplete, xảy ra khi một mẩu tin được sửa đổi thành công trong cơ sở dữ liệu như một kết quả của hoạt động trong điều khiển dữ liệu.

Xóa mẩu tin

Để xóa mẩu tin trong một ứng dụng sử dụng điều khiển dữ liệu, ta dùng phương thức Delete của đối tượng Recordset của điều khiển dữ liệu.

Ví dụ: Adodc1.Recordset.Delete

V.1.6 Dùng sự kiện WillChangeRecord để đảm bảo dữ liệu hợp lệ

Trong lập trình cơ sở dữ liệu, việc đảm bảo rằng dữ liệu nhập vào phù hợp với các quy tắc của một cơ sở dữ liệu người dùng cụ thể là yếu tố quan trọng bậc nhất và mang tính bắt buộc.

Đối với điều khiển ADO Data, việc xác định xem dữ liệu có hợp lệ hay không sẽ được viết trong sự kiện WillChangeRecord của điều khiển. Sự kiện này sẽ được kích hoạt khi người dùng thay đổi thông tin của một mẩu tin và di chuyển sang mẩu tin khác hoặc thêm mới mẩu tin.

Kết nối với cơ sở dữ liệu và làm việc với các mẩu tin thông qua điều khiển Data (DAO Data Control)

Điều khiển Data hạn chế hơn điều khiển ADO Data vì nó chỉ cho phép chúng ta nối kết đến một số nguồn dữ liệu cụ thể, chẳng hạn như Access, Excel, Foxpro,… nhưng Version của các hệ quản trị này cũng là những Version đã lâu đời.

Để sử dụng điều khiển này, ta cần xác định các giá trị của các thuộc tính sau: Connect, DatabaseName, RecordSource.

- Connect xác định cơ sở dữ liệu của ta là loại gì.

- DatabaseName chỉ đến một cơ sở dữ liệu cụ thể.

- RecordSource là một bảng dữ liệu trong cơ sở dữ liệu (đối với cơ sở dữ liệu Access).

Sau khi đã xác định giá trị của các thuộc tính trên thì việc duyệt qua các mẩu tin cũng tương tự như của điều khiển ADO Data.

Thuộc tính EOFAction của điều khiển Data quy định việc chúng ta có thể thêm mới một mẩu tin hay là không (tương tự điều khiển ADO Data).

Các đối tượng truy cập dữ liệu
Mục tiêu: Chương này gồm các bài tập nhằm rèn luyện sinh viên cách thức lập trình cơ sở dữ liệu sử dụng thư viện đối tượng Data Access Objects (DAO). Đây là cách thức lập trình phổ biến đối với các ứng dụng chạy trên máy đơn.

Học xong chương này, sinh viên phải nắm bắt được các vấn đề sau:

- Sử dụng điều khiển dữ liệu để truy xuất cơ sở dữ liệu.

- Sử dụng thư viện đối tượng DAO để lập trình cơ sở dữ liệu.

Kiến thức có liên quan:

- Giáo trình Visual Basic, chương 9.

Tài liệu tham khảo:

Visual Basic 6 Certification Exam Guide - Chapter 5, Page 139 - Dan Mezick & Scot Hillier - McGraw-Hill - 1998.

http://www.vovisoft.com/VisualBasic/VB6Chapter14.htm

http://www.vovisoft.com/VisualBasic/VB6Chapter15.htm

HƯỚNG DẪN

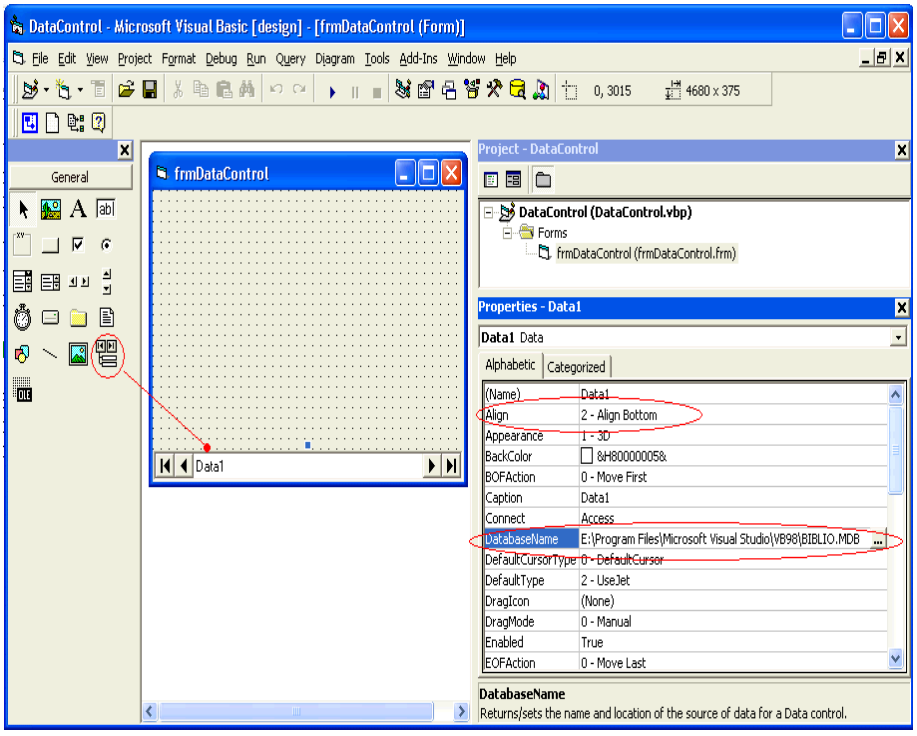
Bài tập 4-1

SỬ DỤNG DATA CONTROL

Bước 1: Tạo một dự án mới tên DataControl trong thư mục Basic\Bt4-1.

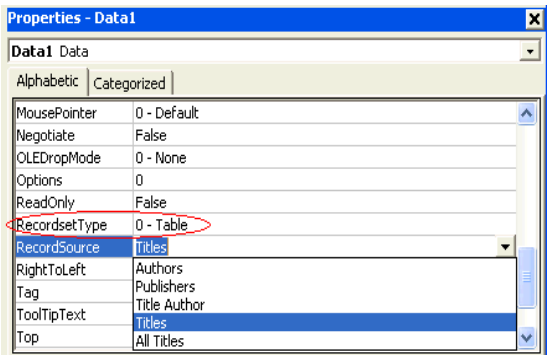
Bước 2: Nhấp đúp lên Icon của Control Data trong Toolbox. Một Control Data tên Data1 sẽ hiện ra trên Form. Muốn cho nó nằm bên dưới Form, hãy đặt thuộc tính Align của nó trong Properties Window thành 2 - Align Bottom.

Nhấp bên phải hàng property DatabaseName, kế đó click lên nút lựa chọn có ba chấm để chọn một file cơ sở dữ liệu Access từ hộp thoại cho Data1. Ở đây ta chọn E:\Program Files\Microsoft Visual Studio\VB98\BIBLIO.MDB (tùy máy tính có thể là ổ C hay ổ đĩa D)



Hình IV.1: Xác lập thuộc tính cho Data Control

Bước 3: Trong chương trình này ta làm việc với table Titles của cơ sở dữ liệu BIBLIO.MDB, để xem và sửa đổi các records. Để ý thuộc tính DefaultType của Data1 có trị số 2- UseJet, tức là dùng kỹ thuật DAO, thay vì dùng kỹ thuật ODBC.

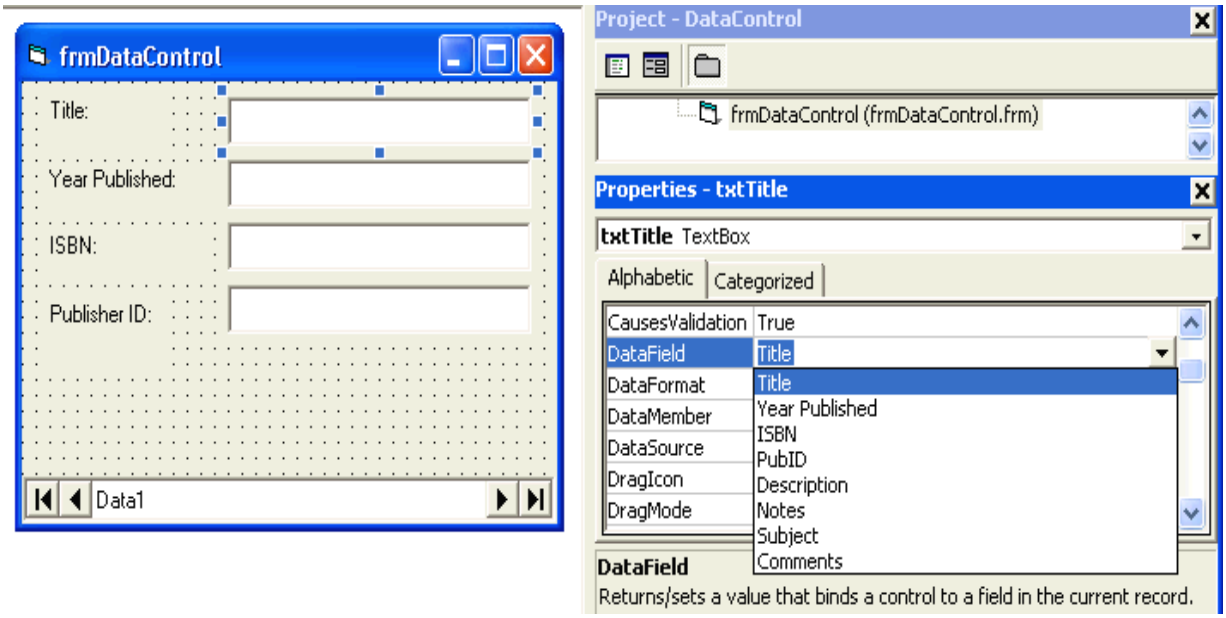


Khi ta nhấp chuột lên thuộc tính Recordsource của Data1, rồi nhấp lên tam giác nhỏ bên phải, một ComboBox sẽ mở ra cho ta thấy danh sách các tables trong cơ sở dữ liệu, chọn Titles. Để ý thuộc tính RecordsetType của Data1 có trị số là 0 - Table:

Hình IV.2: Recordset Type

Bước 4: Một từ mới mà ta sẽ dùng thường xuyên khi truy cập dữ liệu trong VB6 là Recordset (bộ records). Recordset là một Set of records, nó có thể chứa một số records hay không có record nào cả. Một record trong Recordset có thể là một record lấy từ một Table. Trong trường hợp ấy có thể ta lấy về tất cả records trong table hay chỉ những records thỏa đúng một điều kiện, thí dụ như ta chỉ muốn lấy các records của những sách xuất bản trước năm 1990 (Year Published < 1990).

Tạo Form có dạng như sau:



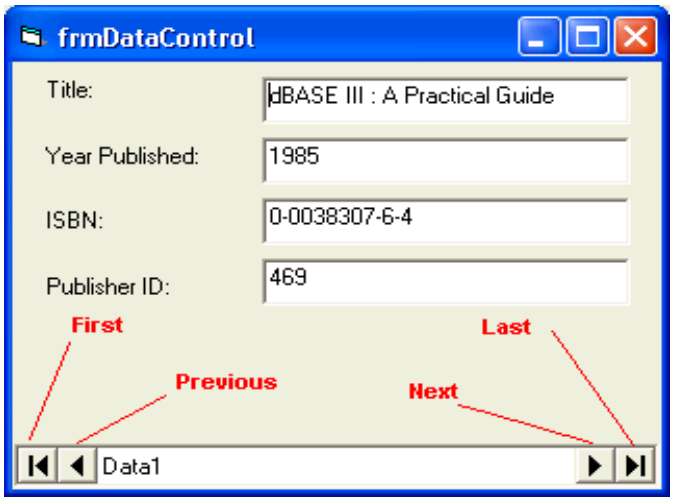
Hình IV.3: Giao diện ban đầu

4 labels với caption của chúng: Title, Year Published, ISBN và Publisher ID. Kế đó cho thêm 4 textboxes tương ứng và đặt tên chúng là txtTitle, txtYearPublished, txtISBN và txtPublisherID.

Bước 5: Chọn textbox txtTitle, rồi đặt thuộc tính Datasource của nó trong Properties Window thành Data1. Khi click lên property Datafield của txtTitle và mở ComboBox ra ta sẽ thấy liệt kê tên các trường trong table Titles. Đó là vì Data1 được coi như trung gian lấy table Titles từ cơ sở dữ liệu. Ở đây ta sẽ chọn cột Title.

Tương tự cho 3 textboxes còn lại, và chọn các cột Year Published (năm xuất bản), ISBN (số lý lịch trong thư viện quốc tế), và PubID (số lý lịch nhà xuất bản) làm Datafield cho chúng.

Bước 6: Lưu dự án và chạy chương trình. Ta sẽ thấy giao diện như sau:



Hình IV.4: Kết quả thực thi ứng dụng

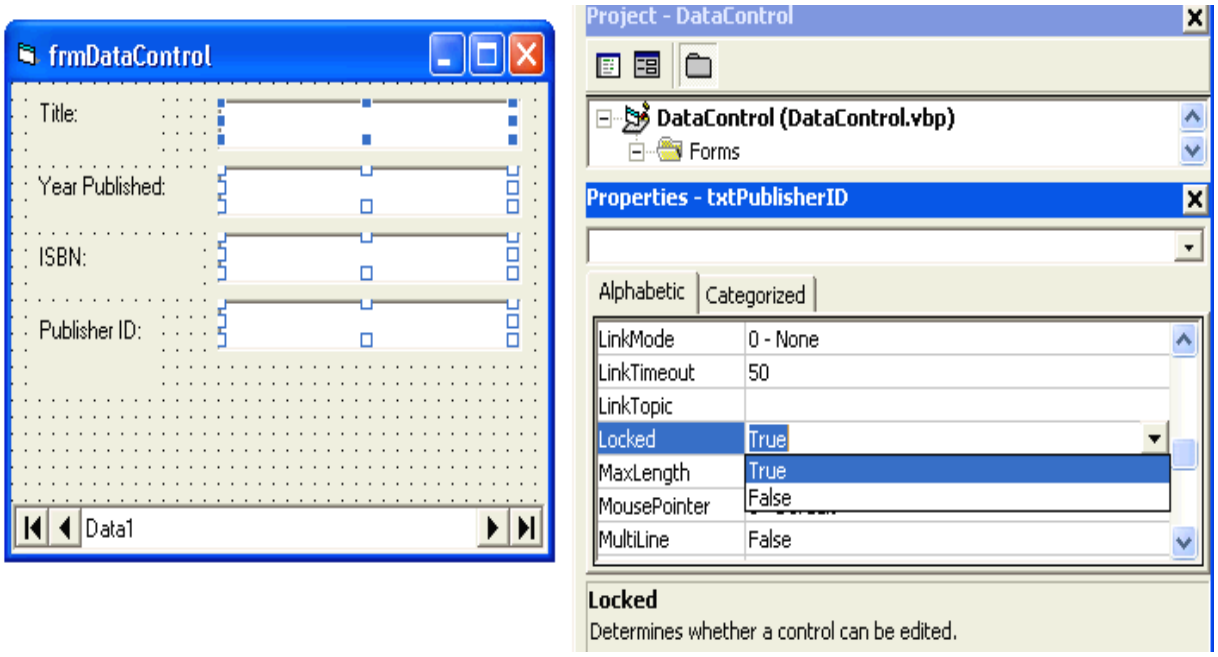
Nhận xét:

- Ta có thể bấm các nút di chuyển Navigator Buttons để đi đến các record đầu (first), trước (previous), kế (next) và cuối (last). Mỗi lần ta di chuyển đến một record mới là chi tiết của record ấy sẽ hiển thị. Nếu không dùng các Navigator Buttons, ta cũng có thể viết đoạn mã để làm công tác tương đương bằng cách gọi các hàm trên Recordset là MoveFirst, MovePrevious, MoveNext và MoveLast.
- Khi record cuối của Recordset đang hiển thị, nếu ta gọi hàm MoveLast thì thuộc tính EOF (End-Of-File) của Recordset trở thành True. Tương tự như vậy, khi record thứ nhất của Recordset đang hiển thị, nếu ta gọi hàm MovePrevious thì thuộc tính BOF (Begin-Of-File) của Recordset trở thành True. Nếu một Recordset không có chứa một record nào cả thì cả hai thuộc tính EOF và BOF đều là True.
- Khi record đầu tiên đang hiển thị, nếu ta sửa Year Published để đổi từ 1985 thành 1983 rồi nhấp Navigator button Next để hiển thị record thứ nhì, kế đó click Navigator button Previous để hiển thị lại record đầu tiên thì ta sẽ thấy là trường Year Published của record đầu tiên đã thật sự được thay đổi (updated) thành 1983.

Điều nầy có nghĩa rằng khi di chuyển từ record nầy đến record khác thì nếu record nầy đã có sự thay đổi do người sử dụng, nó lưu trữ sự thay

đổi đó trước khi di chuyển. Chưa chắc là ta muốn điều nầy, do đó, nếu ta không muốn người sử dụng tình cờ sửa đổi một record thì ta có thể đặt thuộc tính Locked của các textboxes ấy thành True để người sử dụng không thể sửa đổi các textboxes như trong hình dưới đây:

Hình IV.5: Khóa (lock) Textbox



CHỈ ĐỊNH VỊ TRÍ CƠ SỞ DỮ LIỆU LÚC CHẠY CHƯƠNG TRÌNH

Bước 7: Cách chỉ định tên cơ sở dữ liệu trong giai đoạn thiết kế (at design time) ta đã dùng trước đây tuy tiện lợi nhưng hơi nguy hiểm, vì khi ta cài chương trình nầy lên máy tính khác, chưa chắc tập tin cơ sở dữ liệu ấy nằm trong một thư mục có cùng tên. Ví dụ trên máy tính này thì cơ sở dữ liệu nằm trong thư mục E:\Program Files\Microsoft Visual Studio\VB98, nhưng trên máy tính khác thì cơ sở dữ liệu nằm trong thư mục D:\Basic\Bt4-1 chẳng hạn. Do đó, khi chương trình khởi động ta nên xác định lại vị trí của cơ sở dữ liệu. Chẳng hạn ta muốn để cơ sở dữ liệu trong cùng một thư mục với chương trình đang chạy, ta có thể dùng thuộc tính Path của Application Object App.

Khai báo một biến tên duongdan trong phần [General]\[Declaration] của Form1:

Dim duongdan As String

Bước 8: Ta xử lý sự kiện Form_Load như sau:

Private Sub Form_Load()

duongdan = App.Path

If Right(duongdan, 1) <> "\" Then duongdan = duongdan & "\"

Data1.DatabaseName = duongdan & "BIBLIO.MDB"

End Sub

THÊM BỚT CÁC RECORDS

Bước 9: Chương trình đến đây tạm ổn, nhưng nó không cho ta công cụ để thêm (add), bớt (delete) các records. Bây giờ hãy đặt vào Form 5 buttons tên: cmdEdit, cmdNew, cmdDelete, cmdUpdate và cmdCancel.

Bước 10: Lúc chương trình mới khởi động, người sử dụng đang xem thông tin các records thì hai buttons Update và Cancel không cần phải làm việc. Do đó ta sẽ Lock (khóa) các textboxes và disable hai buttons nầy vì không cần dùng chúng.

Bước 11: Trong Sub SetControls dưới đây, ta dùng một tham số gọi là Editing với trị số False hay True tùy theo người dùng đang xem (browse) hay sửa đổi (Edit), ta gọi là Browse mode và Edit mode. Trong Edit mode, các Textboxes được unlocked (mở khóa) và các nút cmdNew, cmdDelete và cmdEdit trở nên vô hiệu lực:

Sub SetControls(ByVal Editing As Boolean)

' Lock/Unlock textboxes

txtTitle.Locked = Not Editing

txtYearPublished.Locked = Not Editing

txtISBN.Locked = Not Editing

txtPublisherID.Locked = Not Editing

' Enable/Disable buttons

CmdUpdate.Enabled = Editing

CmdCancel.Enabled = Editing

CmdDelete.Enabled = Not Editing

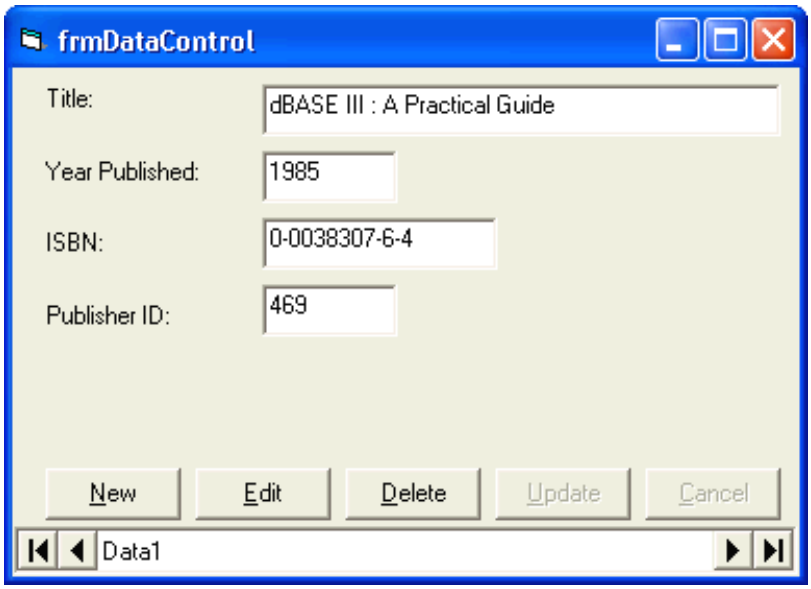
cmdNew.Enabled = Not Editing

CmdEdit.Enabled = Not Editing

End Sub

Trong Browse mode, Form có dạng như sau:

Hình IV.7: Kết quả thực thi



Bước 12: Thủ tục SetControls được gọi trong Sub Form_Load khi chương trình khởi động và sự kiện CmdEdit_Click được xử lý như sau:

```
Private Sub Form_Load()

duongdan = App.Path

If Right(duongdan, 1) <> "\" Then duongdan = duongdan & "\"

Data1.DatabaseName = duongdan & "BIBLIO.MDB"

SetControls (False)

End Sub

Private Sub CmdEdit_Click()

SetControls (True)

End Sub
```

Bước 13: Khi ta xóa một record trong recordset, vị trí của record hiện tại (current record) vẫn không thay đổi. Do đó, sau khi xóa một record ta phải MoveNext. Tuy nhiên, nếu ta vừa xóa record cuối của Recordset thì sau

khi MoveNext, thuộc tính EOF của Recordset sẽ thành True. Thành ra ta phải kiểm tra điều đó, nếu đúng vậy thì lại phải MoveLast để hiển thị record cuối của Recordset như trong đoạn mã của Sub cmdDelete_Click dưới đây:

```
Private Sub CmdDelete_Click()

On Error GoTo DeleteErr

With Data1.Recordset

' Xoa record

.Delete

' Nhay den record ke

.MoveNext

If .EOF Then .MoveLast

Exit Sub

End With

DeleteErr:

MsgBox Err.Description

Exit Sub

End Sub
```

Bước 14: Ta có thể Update (cập nhật) một record trong Recordset bằng hàm Update. Nhưng ta chỉ có thể gọi hàm Update của một Recordset khi Recordset đang ở trong Edit hay AddNew mode. Ta đặt một Recordset vào Edit mode bằng cách gọi hàm Edit của Recordset, thí dụ như Data1.Recordset.Edit. Tương tự như vậy, ta đặt một Recordset vào

AddNew mode bằng cách gọi hàm AddNew của Recordset, thí dụ như Data1.Recordset.AddNew.

Private Sub cmdNew_Click()

Data1.Recordset.AddNew

SetControls (True)

End Sub

Private Sub cmdUpdate_Click()

Data1.Recordset.Edit

Data1.Recordset.Update

SetControls (False)

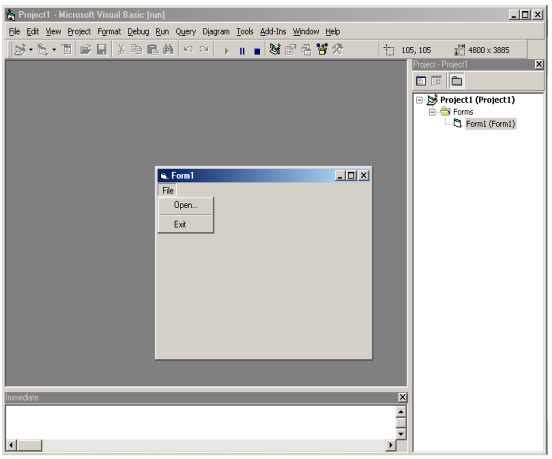
End Sub

Bước 15: Lưu dự án và chạy chương trình.

Bài 4-2

CÁC ĐỐI TƯỢNG CƠ BẢN CỦA DAO

Bước 1: Tạo thư mục Basic\Bt4-2. Tạo giao diện cho chương trình như sau:



Hình IV.8: Giao diện ban đầu

Các tên của thành phần menu lần lượt là: mnuFile, mnuOpen, mnuExit.

Sau đó vào Project\References…, đánh dấu vào Microsoft DAO 3.51 Object Library; chọn OK.

Bước 2: Thêm một Common Dialog vào Form1, tên là dlgDatabase.

Bước 3: Thêm một DBGrid vào form bằng cách chọn: Project\Components…, đánh dấu Microsoft Data Bound Grid Control 5.0 (SP3); rồi chọn DBGrid trên ToolBox. Sau đó thêm một TextBox và một Data Control vào form1. Ta có các tên của điều khiển là: DBGrid1, Text1, Data1 với các thuộc tính như sau:

Item 1: TextBox

Name: Text1

Multiline: True

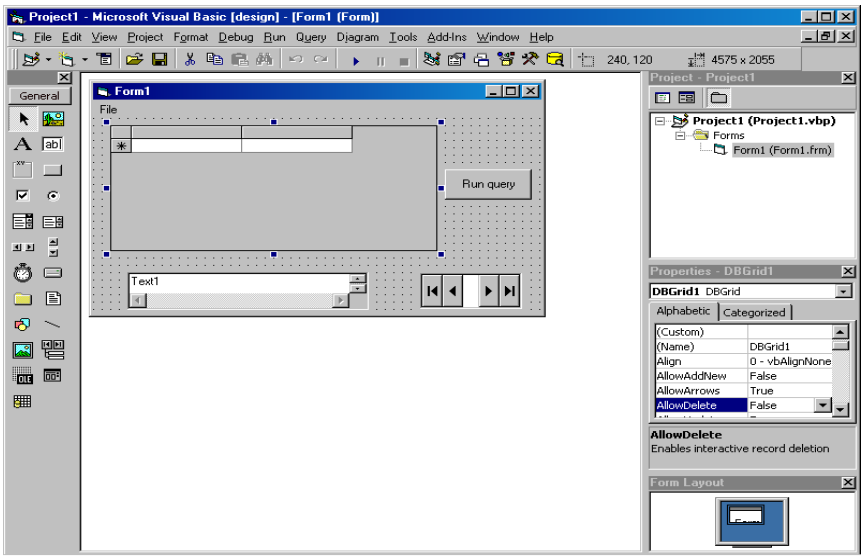
ScrollBars = 3

Item 2: DBGrid

Name: DBGrid1

DataSource = Data1

Ta được hình dạng của form1 như sau:



Hình IV.9: Giao diện đầy đủ

Sau đó, thêm đoạn mã sau trong thủ tục xử lý sự kiện mnuOpen_Click:

CommonDialog1.FileName = "*.mdb"

CommonDialog1.Filter = "Access DBs (*.mdb)|*.mdb"

CommonDialog1.ShowOpen

Data1.DatabaseName = CommonDialog1.FileName

Bước 4: Thêm một nút nhấn (Button) như hình trên, Caption là Run query. Nút này có mục đích là thực thi câu lệnh SQL mà người dùng nhập vào ô Text1. Để thực thi được lệnh SQL này, ta phải gán thuộc tính Recordsource của Data Control Data1 như trong thủ tục xử lý sự kiện Command1_Click:

Private Sub Command1_Click()

Data1.RecordSource = Text1.Text

Data1.Refresh

End Sub

Bước 5: Trong hàm xử lý sự kiện mnuExit_Click thêm dòng mã sau:

End

Chạy chương trình, trong mục File\Open của menu chọn tập tin C:\Program Files\Microsoft Visual Studio\VB98\Biblio.mdb. Sau đó ta gõ câu lệnh SQL sau vào Text Box:

Select * from Publishers

Nhấp chuột vào nút nhấn Run query. Quan sát kết quả hiển thị.

Ta đã tạo một chương trình cho phép người sử dụng để mở một CSDL và chạy câu SQL trên CSDL đó. Bây giờ, đối với CSDL được mở ở trên, tìm xem các bảng của nó là gì nhằm mục đích xây dựng các câu truy vấn cho phù hợp.

Bước 6: Thêm đoạn mã sau vào phần khai báo của Form1:

Private db As DAO.Database

Private td As DAO.TableDef

Private qd As DAO.QueryDef

Private fld As DAO.Field

Bước 7: Trong hàm xử lý sự kiện mnuOpen_Click ta cần kiểm tra xem tập tin được chọn có phải là tập tin CSDL của Access hay không (*.mdb)? Sau đó dùng các biến được khai báo ở trên để thao tác $\Rightarrow$ Sửa thủ tục mnuOpen_Click như dưới đây:

```
Private Sub mnuOpen_Click()
CommonDialog1.FileName = "*.mdb"
CommonDialog1.Filter = "Access DBs (*.mdb)|*.mdb"
CommonDialog1.ShowOpen
If UCase(Right(CommonDialog1.FileName, 3)) <> "MDB" Then
MsgBox "Khong phai la tap tin cua Microsoft Access"
Else
On Error Resume Next
db.Close
On Error GoTo 0
Screen.MousePointer = vbHourglass
' Mo CSDL
Set db = _
DBEngine.Workspaces(0).OpenDatabase(CommonDialog1.FileName)
Form1.Caption = "Cau SQL: Chon " & CommonDialog1.FileName
Screen.MousePointer = vbDefault
Data1.DatabaseName = CommonDialog1.FileName
End If
End Sub
```

Bước 8: Ta đã mở được CSDL, bây giờ ta dùng một List Box để hiển thị tất cả các bảng của CSDL được mở ở trên.

Thêm một ListBox vào Form tên List1, trong hàm xử lý sự kiện mnuOpen, thêm đoạn mã sau trước lệnh End If:

```
' Them vao ListBox

List1.Clear

For Each td In db.TableDefs

List1.AddItem td.Name

Next
```

Chạy chương trình, ListBox sẽ hiển thị tất cả các bảng của CSDL trên.

Bước 9: Thêm một ListBox nữa vào Form, tên List2. Thêm đoạn mã sau trong hàm xử lý sự kiện List1_Click:

```
Private Sub List1_Click()

' Tim bang duoc chon trong CSDL

Set td = New TableDef

For Each td In db.TableDefs

If td.Name = Me.List1.List(Me.List1.ListIndex) Then

Exit For

End If

Next

' Hien thi cac truong cua bang duoc chon
```

```
For Each fld In td.Fields

List2.AddItem fld.Name

Next

End Sub
```

Bước 10: Chạy chương trình, chọn File\Open để chọn tập tin CSDL, lúc đó List1 sẽ hiển thị các bảng của CSDL. Nhấp chọn một bảng trong List1, List2 sẽ hiển thị tên các trường của bảng đó. Bây giờ ta tiến thêm một bước nữa là hiển thị tất cả các câu truy vấn (SQL) được lưu trong CSDL trên bằng cách:

Thêm một ListBox nữa vào Form1 tên là List3, sau đó thêm vào đoạn mã sau trong hàm xử lý sự kiện mnuOpen trước lệnh End If:

```
List2.Clear

List3.Clear

Text1.Text = ""

For Each qd In db.QueryDefs

List3.AddItem qd.Name

Next
```

Bước 11: Chạy chương trình, kiểm tra xem điều gì xảy ra trên List3.

Đóng chương trình lại, thêm đoạn mã sau trong hàm xử lý sự kiện List3_Click:

```
Private Sub List3_Click()

For Each qd In db.QueryDefs

If qd.Name = List3.List(List3.ListIndex) Then
```

```
Text1.Text = qd.SQL

End If

Next

End Sub
```

Chạy chương trình, mở BIBLIO.MDB, nhấp vào List3. Quan sát kết quả.

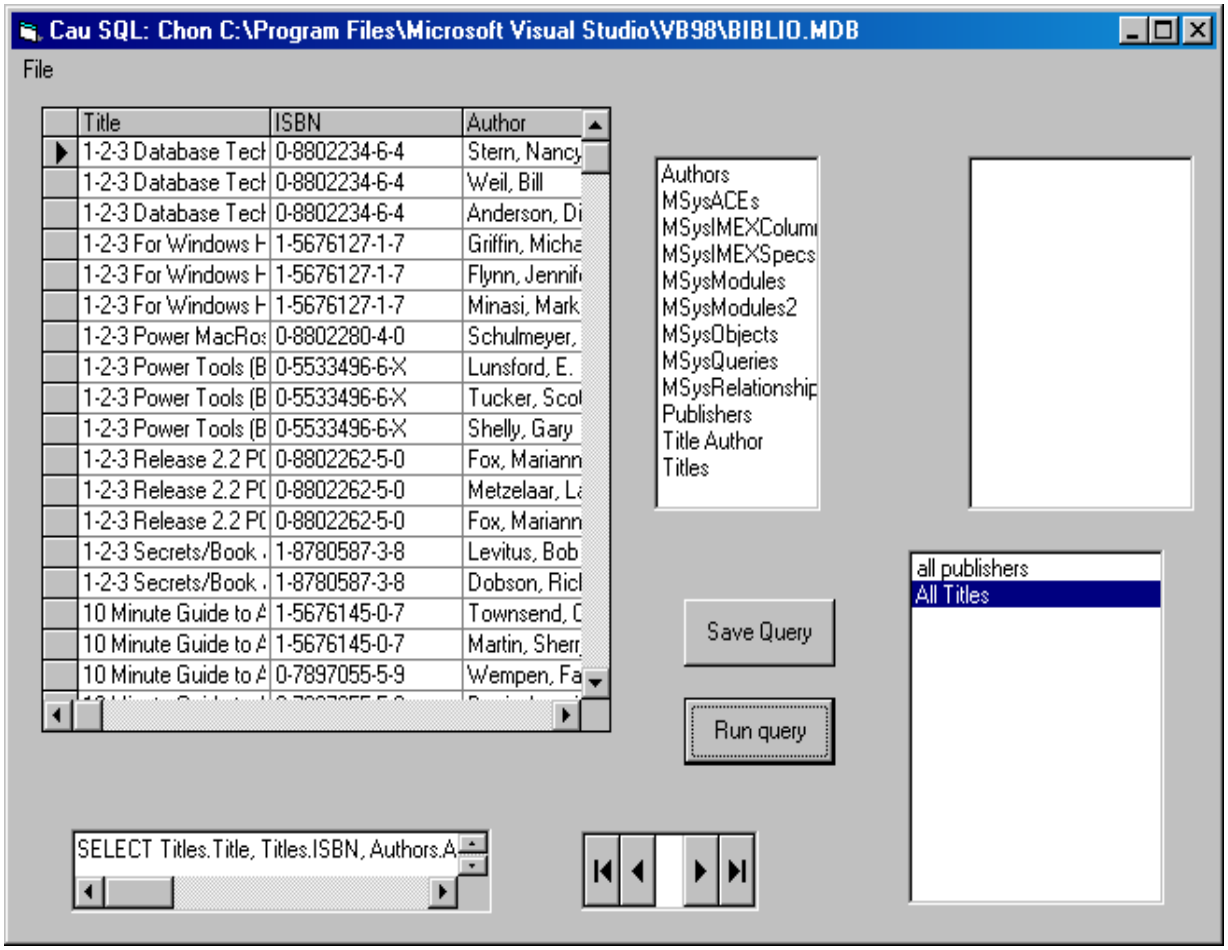
Bước 12: Chúng ta lưu câu SQL nhập từ bàn phím vào trong CSDL trên với một tên cho trước. Ý tưởng chính là ta kiểm tra câu SQL được nhập đó, nếu nó không có lỗi ta sẽ lưu vào CSDL.

Thêm một nút nhấn (Button) vào Form1 với Name: Command2, Caption: Save Query. Sau đó xử lý sự kiện Command2_Click như sau:

```
Private Sub Command2_Click()

' Luu cau SQL

Set qd = New QueryDef

qd.SQL = Trim$(Text1.Text)

MsgBox "Cau SQL duoc luu la: " & qd.SQL

' Nhap ten cua cau SQL

qd.Name = InputBox("Nhap ten cau SQL: ")

db.QueryDefs.Append qd

End Sub
```

Bước 13: Chạy chương trình, mở BIBLIO.MDB, chọn câu một query, chạy nó (Run query); sau đó nhấp vào nút Save Query để lưu lại với tên ta phải nhập vào từ bàn phím. Để kiểm tra, hãy mở lại tập tin trên (File\Open): câu query trên được hiển thị trong List3.

Hình bên dưới hiển thị kết quả khi thực thi chương trình.



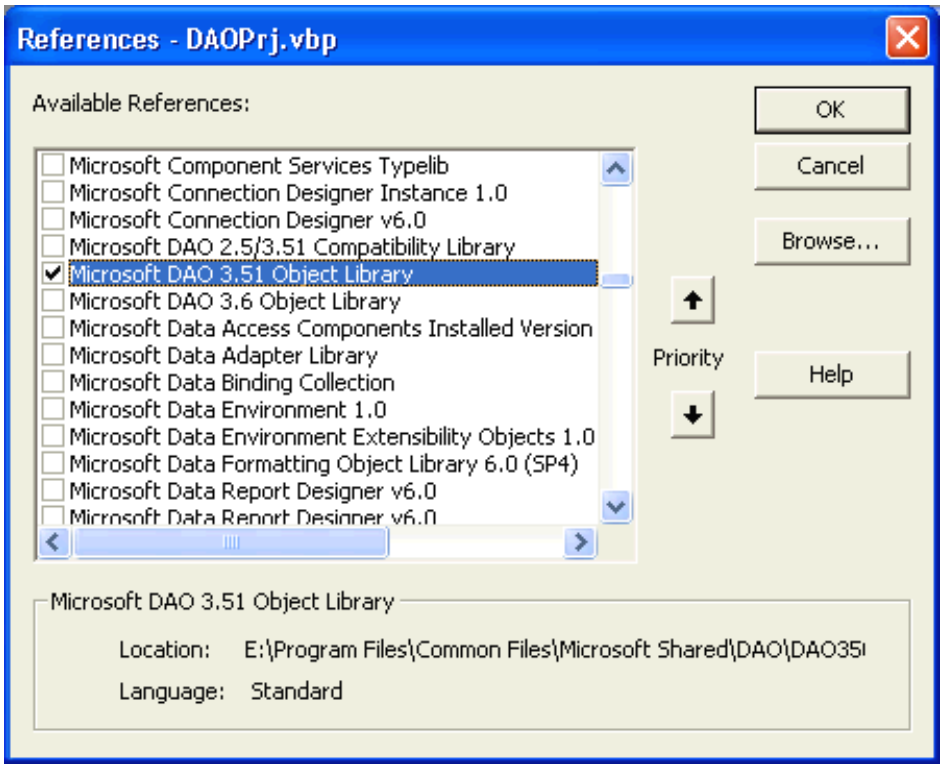
Hình IV.10: Kết quả thực thi ứng dụng

Bài 4-3

MÔ HÌNH DAO

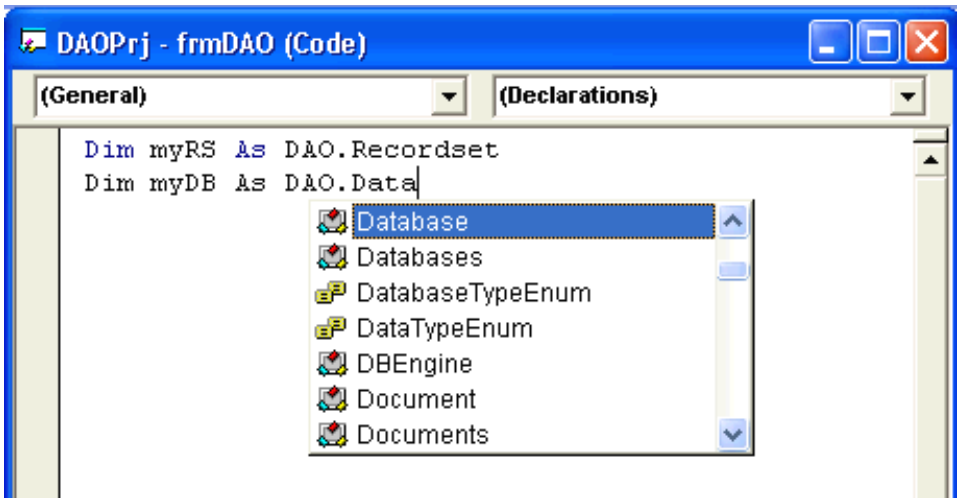
Bước 1: Trong bài này ta sẽ tìm hiểu những cách lập trình căn bản với cơ sở dữ liệu MS Access qua kỹ thuật DAO mà không cần dùng đến Control Data như bài tập 4-1. Ta sẽ cần đến các đối tượng (Object) trong thư

viện DAO, do đó nếu bạn mở một dự án VB mới thì hãy dùng Menu Command Project | References... để chọn Microsoft DAO 3.51 Object Library bằng cách click checkbox bên trái như trong hình dưới đây.



Hình IV.11: Tham chiếu đến thư viện DAO

Bước 2: Sau đó trong cửa sổ soạn thảo mã lệnh của Form chính ta sẽ khai báo biến myDatabase kiểu DAO database và biến myRS cho một DAO recordset. Ở đây ta nói rõ Database và Recordset là thuộc loại DAO để phân biệt với Database và Recordset thuộc loại ADO (ActiveX Data Object) sau này.



Hình IV.12: Khai báo biến

Bước 3: Bây giờ hãy đặt lên Form chính, tên frmDAO, 4 labels với captions: Title, Year Published, ISBN và Publisher ID. Kế đó cho thêm 4 textboxes tương ứng và đặt tên chúng là txtTitle, txtYearPublished, txtISBN và txtPublisherID.

Điều ta muốn làm là khi Form mới được thực thi, nó sẽ lấy về từ cơ sở dữ liệu một Recordset chứa tất cả records trong table Titles theo thứ tự abc của field (trường) Title và hiển thị record đầu tiên.

DÙNG TỪ KHÓA SET

Bước 4: Trước hết là mở một cơ sở dữ liệu dựa vào tên tập tin của Access database:

Set myDB = OpenDatabase(AppFolder & "BIBLIO.MDB")

Để ý từ khóa Set trong đoạn mã trên. Đó là vì myDB là một Pointer (con trỏ) chỉ đến một Object (đối tượng). Mặc dù từ đây về sau ta sẽ dùng myDB như một Database (cơ sở dữ liệu) theo cách giống như bất cứ một biến thuộc kiểu dữ liệu nào khác, nhưng khi chỉ định lần đầu là nó từ đâu đến thì ta dùng chữ Set, để nói rằng thật ra myDB không phải là Object Database, nhưng là Pointer đến Object Database.

Nguyên nhân là VB dành ra một phần trong bộ nhớ (memory) để chứa đối tượng Database khi ta nhận được nó khi hàm OpenDatabase thực thi. Dù vị trí chỗ chứa đối tượng Database trong bộ nhớ không nhất định, nhưng vì ta nắm cán chỉ đến vị trí ấy nên ta vẫn có thể làm việc với nó một cách bình thường. Cái cán ấy là trị số của biến myDB. Vì trị số này không phải là Object (đối tượng), nhưng nó chứa memory address (địa chỉ trong bộ nhớ) chỉ đến (point to) đối tượng Database, nên ta gọi nó là Pointer (con trỏ).

Tương tự như vậy, vì Recordset là một Pointer chỉ đến một đối tượng, ta cũng dùng Set khi chỉ định một DAO Recordset lấy về từ hàm OpenRecordset của database myDB.

```
Set myRS = myDB.OpenRecordset("Select * from Titles ORDER BY Title")
```

Tham số kiểu String ta dùng cho hàm OpenRecordset là một câu lệnh SQL. Nó chỉ định cho cơ sở dữ liệu lấy tất cả mọi trường của mỗi mẩu tin từ Table Titles làm một Recordset và sắp xếp các mẩu tin trong Recordset ấy theo thứ tự abc của trường Title (ORDER BY Title).

Để ý là Recordset nầy cũng giống như thuộc tính Recordset của một Data Control mà ta dùng trong bài 7-1. Bây giờ có Recordset rồi, ta có thể hiển thị chi tiết của record đầu tiên nếu Recordset ấy có ít nhất một record. Ta kiểm tra điều ấy dựa vào thuộc tính RecordCount của Recordset như trong đoạn mã dưới đây của sự kiện Form_Load:

```
Private Sub Form_Load()

AppFolder = App.Path

If Right(AppFolder, 1) <> "\" Then AppFolder = AppFolder & "\"

Set myDB = OpenDatabase(AppFolder & "BIBLIO.MDB")

Set myRS = myDB.OpenRecordset("Select * from Titles ORDER BY Title")
```

```
If myRS.RecordCount > 0 Then

myRS.MoveFirst

Displayrecord

End If

End Sub
```

Bước 5: Sau khi dùng hàm MoveFirst của Recordset để định vị mẩu tin hiện thời là mẩu tin đầu tiên, ta hiển thị trị số các trường của mẩu tin bằng cách gán chúng vào các textboxes của Form như sau:

```
Private Sub Displayrecord()

With myRS

txtTitle.Text = .Fields("Title")

txtYearPublished.Text = .Fields("[Year Published]")

txtISBN.Text = .Fields("ISBN")

txtPublisherID.Text = .Fields("PubID")

End With

End Sub
```

Để ý vì trường Year Published gồm có hai từ nên ta phải đặt tên của trường ấy giữa hai dấu ngoặc vuông ([]). Để tránh bị phiền phức như trong trường hợp nầy, khi đặt tên các trường tcủa table trong lúc thiết kế cơ sở dữ liệu hãy dán dính các chữ lại với nhau, đừng để rời ra. Thí dụ như dùng YearPublished thay vì Year Published.

CÁC NÚT DI CHUYỂN

Bước 6: Muốn có các nút Navigators giống như của một Control Data, ta hãy đặt lên Form 4 buttons mang tên CmdFirst, CmdPrevious, CmNext và CmdLast với captions: <<, <, >, >>.

Bước 7: Mã lệnh cho các nút nầy cũng đơn giản, nhưng ta phải coi chừng khi người dùng muốn di chuyển quá mẩu tin cuối cùng hay mẩu tin đầu tiên. Ta phải kiểm tra xem EOF có trở thành True khi người dùng nhấp CmdNext, hay BOF có trở thành True khi người dùng nhấp CmdPrevious.

Các sự kiện này được xử lý như sau:

```
Private Sub CmdNext_Click()

myRS.MoveNext

If Not myRS.EOF Then

Displayrecord

Else

myRS.MoveLast

End If

End Sub

Private Sub CmdPrevious_Click()

myRS.MovePrevious

If Not myRS.BOF Then

Displayrecord

Else

myRS.MoveFirst
```

```
End If

End Sub

Private Sub CmdFirst_Click()

myRS.MoveFirst

Displayrecord

End Sub

Private Sub CmdLast_Click()

myRS.MoveLast

Displayrecord

End Sub
```

Bước 7: Chạy chương trình. Khi chạy chương trình ta sẽ thấy nó hiển thị chi tiết của mẩu tin đầu tiên khác với các bài trước đây vì các mẩu tin đã được sắp xếp.

Ta hãy thử dùng các nút di chuyển <, <<, >, >> xem chúng làm việc có đúng không.

Tới đây, ta nhận thấy rằng dù người dùng có vô tình sửa đổi một chi tiết nào trong các textboxes, không có mẩu tin nào bị cập nhật hóa trong cơ sở dữ liệu khi người dùng di chuyển từ mẩu tin nầy đến mẩu tin khác. Lý do là các Texboxes không có ràng buộc dữ liệu (Data Bound) với các trường của Recordset.

THÊM BỚT CÁC RECORDS

Bước 8: Giống như chương trình trong bài rồi, ta sẽ thêm công cụ để thêm (add), bớt (delete) các mẩu tin. Hãy thêm vào Form 5 buttons tên:

cmdEdit, cmdNew, cmdDelete, cmdUpdate và cmdCancel.

Bước 9: Chỗ nào trong chương trình 4-1 ta dùng Data1.Recordset thì bây giờ ta dùng myRS.

Ta sẽ dùng lại Sub SetControls với tham số Editing có trị số False hay True tùy theo người dùng đang xem (Browse) hay sửa đổi (Edit). Trong Browse mode, các Textboxes bị Locked (khóa) và các nút cmdUpdate và cmdCancel bị vô hiệu lực. Trong Edit mode, các Textboxes được unlocked (mở khóa) và các nút cmdNew, cmdDelete và cmdEdit bị vô hiệu lực.

Do đó ta chỉ cần nhớ là khi người dùng đang sửa đổi một mẩu tin hiện hành hay thêm một mẩu tin mới. Ta chứa trị số Boolean ấy trong biến AddNewRecord. Nếu user sắp thêm một record mới thì AddNewRecord = True, nếu User sắp Edit một record hiện hữu thì AddNewRecord = False.

Ngoài ra, khi người dùng sắp thêm một mẩu tin mới bằng cách nhấp nút New thì ta phải tự xóa hết các textboxes bằng cách gán chuỗi rỗng cho các TextBox đó.

Ta có các đoạn mã sau:

```
Dim AddNewRecord As Boolean

Private Sub ClearAllFields()

txtTitle.Text = ""

txtYearPublished.Text = ""

txtISBN.Text = ""

txtPublisherID.Text = ""

End Sub

Private Sub cmdNew_Click()
```

AddNewRecord = True

ClearAllFields

SetControls (True)

End Sub

Private Sub CmdEdit_Click()

SetControls (True)

AddNewRecord = False

End Sub

Bước 10: Khi người dùng nhấp Cancel trong khi đang sửa đổi các textboxes, ta không cần gọi hàm vì Recordset chưa bị đặt vào AddNew hay Edit mode. Ở đây ta chỉ cần hiển thị lại chi tiết của mẩu tin hiện hành, tức là hủy bỏ những gì người dùng đang đánh vào:

Private Sub CmdCancel_Click()

SetControls (False)

Displayrecord

End Sub

Bước 11: Lúc người dùng nhấp Update, ta sẽ kiểm tra dữ liệu xem có trường nào bị bỏ trống (nhất là khóa chính ISBN bắt buộc phải có trị số) hay có gì không hợp lệ bằng cách gọi hàm GoodData. Nếu GoodData trả lại một trị số False thì ta không xúc tiến với việc Update. Nếu GoodData trả về trị số True thì ta đặt Recordset vào AddNew hay Edit mode tùy theo trị số của biến AddNewRecord là True hay False.

Giống như khi hiển thị chi tiết của một Record ta phải gán từng trường vào textbox, thì bây giờ khi Update ta phải làm ngược lại, tức là gán nội

dung của từng textbox vào các trường tương ứng. Sau cùng ta gọi hàm Update của recordset và cho các điều khiển trở lại Browse mode:

```
Private Function GoodData() As Boolean

GoodData = True

End Function

Private Sub CmdUpdate_Click()

If Not GoodData Then Exit Sub

With myRS

If AddNewRecord Then

.AddNew

Else

.Edit

End If

.Fields("Title") = txtTitle.Text

.Fields("[Year Published]") = txtYearPublished.Text

.Fields("ISBN") = txtISBN.Text

.Fields("PubID") = txtPublisherID.Text

.Update

End With

SetControls (False)
```

End Sub

TÌM MỘT RECORD

Bước 11: Tiếp theo đây, ta muốn liệt kê các sách có tiêu đề chứa một chữ hay câu nào đó, thí dụ như chữ "Guide". Kế đó người dùng có thể chọn một sách bằng cách chọn tiêu đề sách ấy và nhấp nút Go. Chương trình sẽ locate (tìm ra) record của sách ấy và hiển thị chi tiết của nó.

Bây giờ bạn hãy cho vào Form một textbox tên txtSearch và một Image tên ImgSearch. Kế đó đặt một frame tên fraSearch vào Form. Để lên frame nầy một listbox tên List1 để tựa các sách.

Ta sẽ cho ImgSearch hiển thị hình một ống nhòm nên bạn hãy click vào bên phải property Picture trong Properties Window để chọn Icon BINOCULR.ICO từ folder E:\Program Files\Microsoft Visual Studio\Common\Graphics\Icons\Misc.

Khi người dùng nhấp vào ImgSearch, chương trình sẽ tự động tìm kiếm các sách có tựa được người dùng đánh vào trong TextBox. Sự kiện ImgSearch được xử lý như sau:

```
Private Sub ImgSearch_Click()

fraSearch.Visible = True

Dim SrchRS As DAO.Recordset

Dim SQLCommand As String

SQLCommand = "Select * from Titles where Title LIKE '" & "*" &
txtSearch & "*" & "' ORDER BY Title"

Set SrchRS = myDB.OpenRecordset(SQLCommand)

If SrchRS.RecordCount > 0 Then
```

List1.Clear

With SrchRS

Do While Not SrchRS.EOF

List1.AddItem .Fields("Title")

.MoveNext

Loop

End With

End If

End Sub

Trong câu SELECT trên ta dùng toán tử LIKE, nội dung của TextBox, có dấu * ở hai bên. Dấu * là chỗ có (hay không có) chữ gì cũng được.

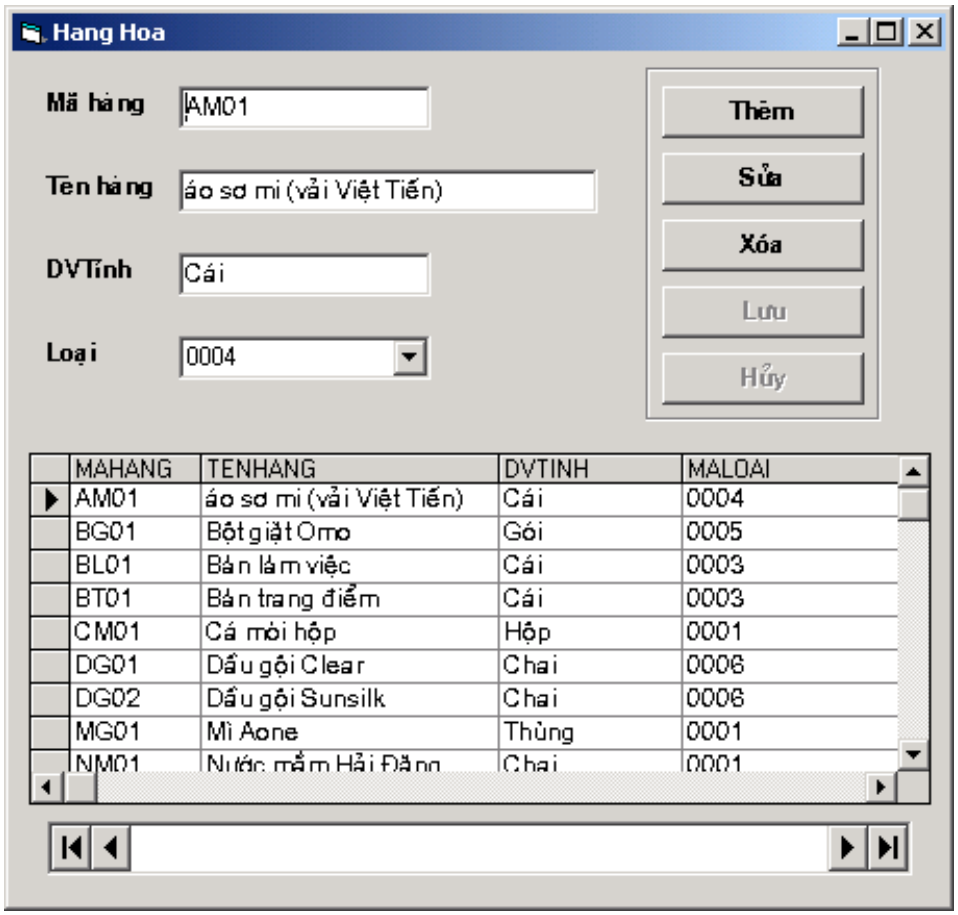
Bước 12: Lưu dự án và chạy chương trình. Kiểm tra kết quả.

Bài tập 4-4

THÍ DỤ VỀ SỬ DỤNG DAO

Bài tập này nhằm mục đích giới thiệu về cách thức sử dụng điều khiển dữ liệu và thư viện DAO trong việc thiết kế một Form nhập liệu hoàn chỉnh cho bảng THangHoa trong CSDL HangHoa.mdb. Giao diện cùng với mã lệnh chỉ mang tính chất gợi ý; sinh viên có thể thực hiện theo ý riêng của mình.

Bước 1: Thiết kế form như sau:



Hình IV.13: Sử dụng Data Control

+

*

Với *: DataControl: Điều khiển dữ liệu

DatabaseName: HangHoa.MDB

RecordSource: THangHoa

Name: datHH

Bước 2: Thiết lập các thuộc tính cho các TextBox & ComboBox.

DataSource: DatHH

DataField:

Bước 3: Đặt điều khiển lưới lên Form (+) sau khi đã tham chiếu đến nó

(Project\Components\Microsoft Data Bound Grid Control 5.0 SP3).

DataSource: datHH

Chạy chương trình, ta được kết quả như trên.

Bước 4: Sự kiện cmdThem_Click

```
Private Sub cmdThem_Click()
datHH.Recordset.AddNew
DoiTThai False
End Sub
```

Trong đó:

```
Private Sub DoiTThai(ByVal TThai As Boolean)
cmdThem.Enabled = TThai
cmdSua.Enabled = TThai
cmdXoa.Enabled = TThai
cmdHuy.Enabled = Not TThai
cmdLuu.Enabled = Not TThai
End Sub
```

Bước 4: Sự kiện cmdSua_Click & sự kiện cmdXoa_Click:

```
Private Sub cmdSua_Click()
datHH.Recordset.Edit
DoiTThai False
End Sub
Private Sub cmdXoa_Click()
On Error GoTo Xuly
datHH.Recordset.Delete
Exit Sub
Xuly:
MsgBox Err.Description, vbCritical + vbSystemModal, "Loi"
End Sub
```

Bước 5: Sự kiện cmdLuu_Click & cmdHuy_Click

```
Private Sub cmdHuy_Click()
'dathh.Database =
datHH.Recordset.CancelUpdate
DoiTThai True
End Sub
Private Sub cmdLuu_Click()
```

datHH.Recordset.Update

DoiTThai True

End Sub

Bước 6: Sự kiện Form_Load:

Private Sub Form_Load()

DoiTThai True

End Sub

- Có nhận xét gì nếu khi chạy chương trình, ta thêm mới một mẩu tin có khóa là Mahang trùng với một MaHang đã có. Để giải quyết ta làm thế nào?

BÀI TẬP TỰ LÀM

Sử dụng CSDL HangHoa.mdb, anh (chị) hãy:

1. Cải tiến Form nhập ở bài 4-4, sao cho trường MaLoai phải được lấy từ các MaLoai của bảng TLoaiHang. Hơn nữa thay vì hiển thị MaLoai, ta hiển thị TenLoai cho dễ theo dõi; nhưng khi thêm vào bảng THangHoa, ta lại thêm vào MaLoai của TenLoai đó.

Bằng cách sử dụng DAO (tương tự 4-3), anh (chị) hãy:

- Thiết kế Form nhập liệu cho bảng THANGHOA.
- Thiết kế Form nhập liệu cho bảng TNHANVIEN.

ODBC và các đối tượng dữ liệu từ xa
Mục tiêu: Chương này giới thiệu về thư viện đối tượng Remote Data Objects, cách thức được sử dụng để truy cập các đối tượng dữ liệu từ xa.

Học xong chương này, sinh viên phải nắm bắt được các vấn đề sau:

- Khái niệm Open Database Connectivity (ODBC).

- Sử dụng điều khiển dữ liệu từ xa (Remote Data Control) để truy cập dữ liệu.

- Cây phân cấp của mô hình đối tượng RDO.

- Sử dụng thư viện đối tượng RDO để tương tác với cơ sở dữ liệu trong VB.

Kiến thức có liên quan:

- Các cấu trúc lập trình trong VB.

- Câu lệnh truy vấn dữ liệu trong cơ sở dữ liệu.

- Nắm bắt được các mô hình DAO là một lợi thế vì lúc đó việc tiếp thu mô hình ADO được nhanh hơn.

Tài liệu tham khảo:

- Microsoft Visual Basic 6.0 & Lập trình cơ sở dữ liệu – Chương 23, trang 735 - Nguyễn Thị Ngọc Mai (chủ biên), Nhà xuất bản Giáo dục - 2000.

- Tự học Lập trình cơ sở dữ liệu với Visual Basic 6 trong 21 ngày (T2) – Chương 17, trang 227 - Nguyễn Đình Tê (chủ biên), Nhà xuất bản Giáo dục - 2001.

Open Database Connectivity (ODBC)

Khái niệm

ODBC là công nghệ Windows cho phép ứng dụng Client nối với cơ sở dữ liệu từ xa. Nằm trên máy Client, ODBC làm cho nguồn dữ liệu quan hệ trở nên trong suốt đối với ứng dụng Client. Vì thế ứng dụng Client không cần quan tâm đến kiểu cơ sở dữ liệu là gì.

ODBC gồm 3 phần:

- Trình quản lý điều khiển (driver manager).

- Một hay nhiều trình điều khiển (driver).

- Một hay nhiều nguồn dữ liệu (data source).

Kiến trúc

Kiến trúc ODBC chứa kết nối giữa ứng dụng Client và cơ sở dữ liệu Server thông qua trình quản lý điều khiển ODBC.

Ứng dụng ClientNguồn dữ liệu ODBCTrình quản lý điều khiển ODBCTrình điều khiển ODBCDBHình 10.1: Kiến trúc ODBC trình bày kết nối giữa ứng dụng Client và CSDL Server thông qua trình quản lý điều khiển ODBC

Tạo nguồn dữ liệu ODBC

Để một ứng dụng Client nối với cơ sở dữ liệu Client/Server dùng ODBC; trước hết, ta phải cung cấp thông tin về nguồn dữ liệu ODBC trên Client. Mỗi Server yêu cầu những gói thông tin khác nhau để nối với Client. ODBC cung cấp cho thông tin này một tên đơn giản để ta có thể tham chiếu đến nó, thay vì phải thiết lập gói thông tin từ đầu mỗi lần ta cần đến nó.

Ứng dụng Client có thể tham chiếu một cách dễ dàng đến tổ hợp của một điều khiển, một cơ sở dữ liệu và có thể thêm một người sử dụng và mật khẩu. Tên này chính là tên nguồn dữ liệu hay Data Source Name (DSN).

Để tạo một tên nguồn dữ liệu ODBC trên máy Client, ta làm như sau:

- Mở Control Panel.
- Chọn Administrative Tools\Data Source (ODBC), hộp thoại quản trị nguồn dữ liệu xuất hiện:

Hình 10.2: Hộp thoại quản trị nguồn dữ liệu ODBC

[missing_resource: .png]

- Ta có thể tạo một trong ba kiểu nguồn dữ liệu ODBC:
    - User DSN: chỉ có người dùng tạo ra nó mới có thể sử dụng (trên máy đang dùng).
    - System DSN: bất kỳ ai sử dụng máy này đều có thể dùng được. Đây cũng là kiểu nguồn dữ liệu mà ta cần tạo khi cài đặt ứng dụng cơ sở dữ liệu Web.
    - File DSN: có thể được copy và sử dụng bởi máy khác.

- Khi hộp thoại ODBC đã mở ra, chọn lớp UserDSN (hay System DSN), Tạo một kết nối mới, nhấn nút Add, màn hình sẽ hiện ra như sau:

[missing_resource: .png]

Hình 10.3: Lựa chọn loại cơ sở dữ liệu cần thiết để tạo kết nối

- Chọn loại CSDL mà ta muốn thao tác (Access, Foxpro, SQL Server,…), nhấn Finish. Sau khi nhấn Finish, một màn hình sẽ hiện ra cho phép ta nhập vào Data Source Name, đây là tên của kết nối CSDL. Tên của kết nối không cần phải giống với tên của cơ sở dữ liệu. Phần Description dùng để gõ các thông tin mô tả về kết nối. Ngoài ra ta còn phải chọn đường dẫn đến tập tin CSDL tương ứng.

Truy cập nguồn dữ liệu với điều khiển DAO Data và ODBC Direct

DAO tự động nạp bộ máy cơ sở dữ liệu Jet mỗi khi nó truy cập dữ liệu Client/Server, thậm chí khi ta không thực sự sử dụng cơ sở dữ liêu Jet/Access.

Ngoài ra ta còn có thêm tùy chọn sử dụng ODBCDIRECT để truy cập dữ liệu Client/Server. Đây là một chuyển đổi để truy cập cơ sở dữ liệu server trực tiếp thông qua DAO mà không cần nạp bộ máy cơ sở dữ liệu Jet. Tùy chọn này thích hợp khi ta phải dùng DAO để truy cập dữ liệu Client/Server mà không cần bận tâm về tính linh hoạt, khả năng sử dụng lại và tính dễ bảo trì của chương trình.

Ví dụ: Tạo một Form có sử dụng ODBCDirect và DAO Data Control.

- Tạo dự án mới.
- Tham chiếu đến điều khiển lưới Microsoft Data Bound Grid Control trong mục Project\Components.
- Tạo Form có dạng sau:

[missing_resource: .png]

12Hình 10.4: Ví dụ về ODBC Direct

1: DBGrid. Name: dbGrid1.

2: Data Control. Name: Data1.

- Đổi thuộc tính DefaultType của DataControl là 1 – Use ODBC. Điều này làm cho chương trình thực hiên nhanh hơn.
- Thuộc tính Connect của Data1 là: ODBC;DSN=DBHH
- Đặt thuộc tính Record Source của Data Control:

Select * From THANGHOA

Đặt thuộc tính Data Source của DBGrid1 là: Data1.

Lưu dự án và thực thi chương trình.

Truy cập dữ liệu dùng điều khiển dữ liệu từ xa

Điều khiển dữ liệu từ xa (Remote Data Control - RDC) là một cách khác truy cập dữ liệu từ xa trong các ứng dụng Visual Basic. Điều khiển này dùng một giao diện lập trình tương tự như điều khiển ADO Data hay DAO Data.

Ví dụ: Sử dụng RDC để hiển thị dữ liệu trên lưới:

- Ở đây ta có sử dụng Remote Data Control và lưới hiển thị dữ liệu, do đó ta tham chiếu đến các thành phần này bằng cách chọn Project\Components…, thiết lập tham chiếu đến Microsoft Remote Data Control và Microsoft Data Bound Grid Control. Nhấp OK.
- Thiết kế Form có dạng sau:

[missing_resource: .png]

11Hình 10.5: Ví dụ về Remote Data Control

1: RemoteDataControl. Name: rdcHangHoa

2: DBGrid. Name: dbgHangHoa.

- Đặt thuộc tính DataSourceName của điều khiển rdcHangHoa là DBHH (DSN đã tạo trước đây).
- Định thuộc tính SQL của điều khiển rdcHangHoa là:

Select * From THANGHOA

- Chỉ định thuộc tính DataSource của điều khiển dbgHangHoa là rdcHangHoa.
- Thực thi chương trình.

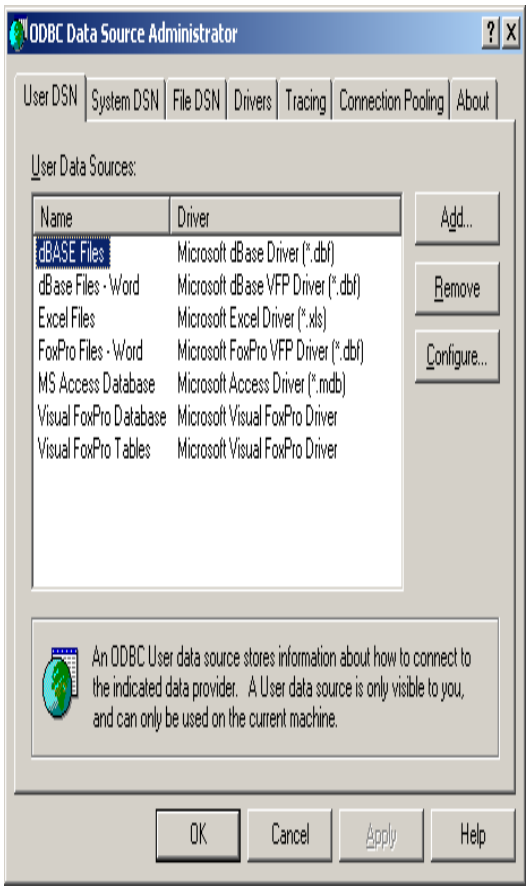
Remote Data Object (RDO)

Đối tượng dữ liệu từ xa (Remote Data Objects - RDO) được mô tả dạng hình cây tương tự như DAO. Tuy nhiên mô hình đối tượng RDO đơn giản hơn DAO.

Dù vậy ta không thể dùng RDO để tạo đối tượng cơ sở dữ liệu như bảng, view, thủ tục chứa sẵn…

Để tham chiếu đến RDO ta vào Project\References…\Microsoft Remote Data Object 2.0.

Mô hình đối tượng RDO:



Đối tượng RDOEngine

Đây là đối tượng ở cấp cao nhất trong mô hình RDO, giới thiệu về các thành phần của mô hình. Ta không cần phải khởi tạo đối tượng này cách tường minh bởi vì nó được khởi tạo tự động.

Đối tượng rdoEngine cho phép ta truy cập toàn bộ các đối tượng khác trong mô hình RDO. Ngoài ra, nó còn được sử dụng để tạo mới một Data Source Name (DSN) nhờ vào phương thức rdoRegisterDataSource. Đoạn mã lệnh dưới đây cho phép đăng ký một nguồn dữ liệu mới cho cơ sở dữ liệu Pubs (của SQL Server):

```
Dim strTemp As String

strTemp = “Description=Test DSN” & Chr(13) & _

“SERVER=(local)” & Chr(13) & “Database=Pubs”

rdoEngine.rdoRegisterDataSource “MyTestPubs”, “SQL Server”, _

True, strTemp
```

Đối tượng RDOError

Đối tượng lỗi xác định thao tác nào trên một nguồn dữ liệu gây ra lỗi. Đây không phải là các lỗi của Visual Basic vì nó xảy ra đối với cơ sở dữ liệu. Vì thế, ta không cần phải bẫy lỗi cho các lỗi này. Thay vào đó, tập hợp rdoErrors sẽ xác định thao tác của ta là thành công hay thất bại. Chẳng hạn như đoạn mã lệnh sau:

```
Dim objError As RDO.rdoError

Dim strError As String

For Each objError In rdoEngine.rdoErrors

strError = strError & objError.Description & vbCrLf

Next
```

```
' Display Errors

MsgBox "The following errors occurred: " & strError
```

Đối tượng RDOEnvironment

Đối tượng này chỉ ra cách thức bảo mật của cơ sở dữ liệu. Ta sử dụng đối tượng này để xác định định danh người dùng cùng mật khẩu hay thi hành một phiên giao dịch (Transation) trên cơ sở dữ liệu.

Đối tượng này không được khởi tạo trực tiếp, chúng được tạo ra tự động.

Đối tượng RDOConnection

Phần lớn các chức năng của RDO bắt đầu với đối tượng rdoConnection; đối tượng này cho phép thiết lập một nối kết đến một nguồn dữ liệu ODBC. Khi nguồn dữ liệu đã được định nghĩa, các thuộc tính cùng các phương thức của đối tượng này được sử dụng để giao tiếp với nguồn dữ liệu ấy.

Đoạn mã lệnh đơn giản dưới đây cho phép tạo một nối kết đến nguồn dữ liệu có tên là Biblio.

```
Set m_Connection = New RDO.rdoConnection

m_Connection.Connect = “DSN=Biblio”

m_Connection.EstablishConnection
```

Đối tượng RDOQuery

Đối tượng này được sử dụng để thực thi các câu truy vấn trên nguồn dữ liệu ODBC. Các câu truy vấn này có thể là các câu SQL hay là lời gọi thực thi các thủ tục lưu trữ sẵn trong cơ sở dữ liệu. Nếu là lời gọi các

thủ tục lưu trữ sẵn thì tham số của các thủ tục này được xác định nhờ đối tượng rdoParameter.

Hai yếu tố then chốt để xác định một đối tượng rdoQuery là nối kết nào cần truy vấn và câu lệnh SQL để truy vấn dữ liệu.

Ví dụ: Giả sử ta có đoạn mã lệnh bên trên để tạo nối kết đến nguồn dữ liệu Biblio. Đoạn mã lệnh dưới đây thực thi câu lệnh SQL lấy về thông tin về tất cả các nhà xuất bản:

```
‘ Tạo câu SQL

Dim strSQL As String

strSQL = “SELECT * FROM Publishers”

‘ Tạo đối tượng Query

Dim m_Query As RDO.rdoQuery

Set m_Query = New RDO.rdoQuery

Set m_Query.ActiveConnection = m_Connection

m_Query.SQL = strSQL

m_Query.Excute
```

Đối tượng RDOResultset

Đối tượng này quản lý các mẩu tin được trả về từ một nối kết qua ODBC. Tập các mẩu tin có thể được trả về thông qua đối tượng rdoQuery hay nhờ phương thức OpenResultset của một đối tượng rdoConnection. Trong một số trường hợp, tập các mẩu tin được truy xuất nhờ vào thuộc tính LastQueryResults của đối tượng rdoConnection:

```
Set m_Resultset = m_Connection.LastQueryResults
```

Khi đối tượng rdoResultset được tạo ra, ta có thể di chuyển đến mẩu tin xác định bằng các phương thức MoveFirst, MoveLast, MoveNext, MovePrevious.

Đối tượng RDOTable

Đối tượng này xác định một bảng trong một nguồn dữ liệu ODBC. Đối tượng này không được sử dụng cho các thao tác truy xuất dữ liệu mà nó chỉ được sử dụng khi ta cần xác định cấu trúc các bảng của cơ sở dữ liệu của ta.

Đối tượng RDOParameter

Đối tượng này xác định các tham số đầu vào hay các kết quả nhận được từ các thủ tục lưu trữ sẵn trong cơ sở dữ liệu.

Sử dụng RDO trong chương trình

Thiết lập kết nối đến nguồn dữ liệu

- Trước tiên ta phải tham chiếu đến mô hình đối tượng RDO.
- Nối kết đến nguồn dữ liệu nhờ đối tượng rdoConnection.

- Khai báo một biến đối tượng rdoConnection.

- Khởi tao đối tượng, sau đó thiết lập các thuộc tính và các phương thức thích hợp để hoàn tất kết nối; trong đó có hai thuộc tính cần quan tâm là chuỗi kết nối (ConnectionString) và loại con trỏ (Cursor Driver).

- Thiết lập nối kết nhờ phương thức EstablishConnection của đối tượng rdoConnection.

Chuỗi kết nối (ODBC Connect String)

Chuỗi kết nối được sử dụng khi thiết lập nối kết đến nguồn dữ liệu. Đây là một chuỗi (String) xác định các thông tin quan trọng gởi đến trình điều khiển ODBC để truy cập dữ liệu. Các thông tin cần quan tâm: tên nguồn dữ liệu (DSN Name), User ID, Password.

Các tham số của chuỗi kết nối ODBC:

Tham số	Ý nghĩa
DSN	Tên nguồn dữ liệu ODBC
UID	Tên người dùng cơ sở dữ liệu
PWD	Mật khẩu truy cập
DRIVER	Trình điều khiển DBC
DATABASE	Tên của cơ sở dữ liệu được nối kết
SERVER	Tên máy chứa cơ sở dữ liệu phục vụ (database server)
WSID	Tên máy chứa cơ sở dữ liệu khách (database client)
APP	Tên của tập tin chương trình (*.exe)

Chuỗi kết nối được xác lập nhờ thuộc tính Connect của đối tượng rdoConnection.

Ví dụ: Chuỗi nối kết đến 1 DSN của cơ sở dữ liệu Access Biblio:

Set m_Connection = New RDO.rdoConnection

m_Connection.Connect = "DSN=Biblo"

m_Connection.EstablishConnection

Chuỗi nối kết đến DSN Publications của cơ sở dữ liệu SQL Server:

Set m_Connection = New RDO.rdoConnection

m_Connection.Connect = "DSN=Biblo;UID=sa;PWD=;"

m_Connection.EstablishConnection

Bên cạnh nối kết chuẩn thông qua ODBC, RDO cũng hỗ trợ loại nối kết DSN cấp thấp (DSN-less connection). Đối với loại này, ta không cần định nghĩa tên nguồn dữ liệu (DSN) trong bộ quản trị ODBC của Windows. Lúc này tất cả các thông tin về cơ sở dữ liệu được cung cấp đầy đủ trong chuỗi nối kết.

Ví dụ: Chuỗi nối kết cấp thấp đến cơ sở dữ liệu SQL Server Pubs:

Set m_Connection = New RDO.rdoConnection

m_Connection.Connect ="DRIVER={SQL Server};” & _

“SERVER=(local);DATABASE=Pubs;UID=admin;PWD=;DSN=;"

m_Connection.EstablishConnection

Trình điều khiển con trỏ (Cursor Drivers)

Trình điều khiển con trỏ xác định cách thức tập các mẩu tin trả về từ cơ sở dữ liệu được lưu trữ như thế nào, có thể chúng được lưu trữ tại máy chủ (server) hay máy trạm (client).

Trình điều khiển con trỏ được thiết lập nhờ thuộc tính CursorDriver của đối tượng Connection (trước khi gọi thực thi phương thức EstablishConnection) và các giá trị sau có thể đề cử cho nó:

Giá trị	Ý nghĩa
rdUseIfNeeded	Trình điều khiển ODBC tự động xác định loại con trỏ (được lưu phía server hay client). Nếu có thể, con trỏ loại được lưu phía server được đề cử.
rdUseODBC	Xác định loại con trỏ của ODBC chuẩn, nghĩa là tập tất cả các mẩu tin được lưu ở máy client.
rdUseServer	Tập các mẩu tin được lưu phia server. Tuy nhiên loại con trỏ này không thích hợp lắm trong môi trường đa người dùng.
rdUseClientBatch	Giống như rdUseODBC nhưng có thể được cập nhật đồng thời.
rdUseNone	Không sử dụng con trỏ, tập các mẩu tin được trả về một lần duy nhất lúc chúng được yêu cầu. Ta chỉ có thể di chuyển tới trong tập các mẩu tin kết quả.

Đối tượng dữ liệu ActiveX
Mục tiêu: Chương này giới thiệu về thư viện ActiveX Data Object (ADO), thư viện đối tượng được sử dụng nhiều nhất trong các ứng dụng truy cập cơ sở dữ liệu dạng khách/chủ (Client/Server) hiện nay.

Học xong chương này, sinh viên phải nắm bắt được các vấn đề sau:

- Kiến trúc OLEDB/ADO.

- Cây phân cấp của mô hình đối tượng ADO.

- Sử dụng thư viện đối tượng ADO để tương tác với cơ sở dữ liệu trong VB.

Kiến thức có liên quan:

- Các cấu trúc lập trình trong VB.

- Câu lệnh truy vấn dữ liệu trong cơ sở dữ liệu.

- Nắm bắt được các mô hình DAO, RDO là một lợi thế vì lúc đó việc tiếp thu mô hình ADO được nhanh hơn.

Tài liệu tham khảo:

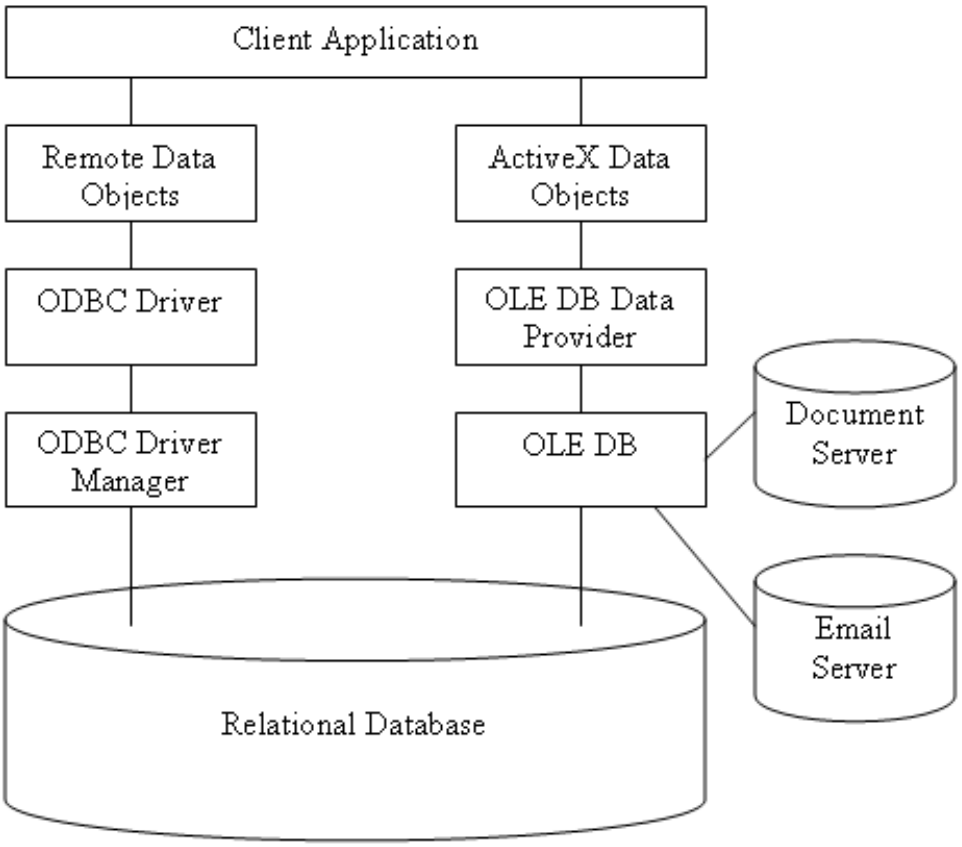
- Microsoft Visual Basic 6.0 & Lập trình cơ sở dữ liệu - Chương 27, trang 877 - Nguyễn Thị Ngọc Mai (chủ biên) – Nhà xuất bản Giáo dục - 2000.

- Tự học Lập trình cơ sở dữ liệu với Visual Basic 6 trong 21 ngày (T2) – Chương 18, trang 277 - Nguyễn Đình Tê (chủ biên) - Nhà xuất bản Giáo dục - 2001.

ADO (ActiveX Data Objects) là công nghệ truy cập cơ sở dữ liệu hướng đối tượng tương tự như DAO. Hiện nay, ADO được Microsoft xem kỹ thuật chính để truy cập dữ liệu từ Web Server.

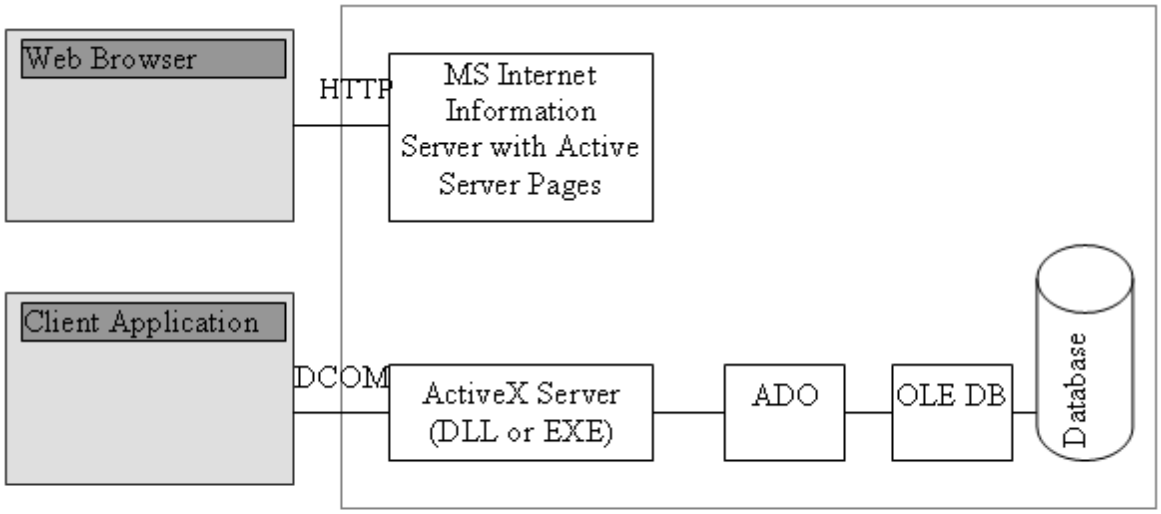
Kiến trúc OLE DB/ADO

ADO sử dụng OLEDB như là trình cung cấp dữ liệu cơ sở. Trình cung cấp OLE DB cho phép người lập trình có thể truy xuất dữ liệu từ cả hai nguồn: quan hệ và phi quan hệ. VB6.0 đã hỗ trợ các trình cung cấp cục bộ cho SQL Server, Oracle và Microsoft Jet/Access.



Hình 11.1: Mô hình lập trình CSDL Client - Server dùng RDO và ADO

Ta chỉ cần lập trình với phần giao diện người sử dụng ở phía Client. Việc truy cập cơ sở dữ liệu trên trình duyệt Web hay ứng dụng VB được thực hiện nhờ ADO. Cấu trúc này cho phép ta lập trình một cách nhất quán trên Web cũng như trên ứng dụng.



Hình 11.2: Truy cập CSDL từ trình ứng dụng & trình duyệt WEB theo ADO

Mô hình ADO

Mô hình ADO được trình bày theo dạng phân cấp (tương tự DAO và RDO).

Để có thể lập trình với thư viện ADO, ta phải tham chiếu đến thư viện này bằng cách chọn Project\References…\Microsoft ActiveX Data Object 2.0.

[missing_resource: .png]

Hình 11.3: Mô hình ADO

Mô hình ADO có 3 đối tượng cốt lõi:

- Connection: kết nối CSDL thật sự.
- Command: thực thi các câu truy vấn dựa vào kết nối dữ liệu.
- RecordSet: là tập các mẩu tin được chọn từ câu truy vấn thông qua đối tượng Command.

Các đối tượng trong mô hình ADO

Đối tượng Connection

Dùng phương thức Open của đối tượng Connection để thiết lập kết nối với nguồn dữ liệu. Để thực hiện điều này ta cần phải thông báo với ADO thông tin kết nối với dạng chuỗi theo kiểu chuỗi kết nối của ODBC. Thuộc tính ConnectionString thực hiện điều này. Ngoài ra ta còn có thể chọn trình cung cấp bằng cách quy định giá trị của thuộc tính Provider của đối tượng.

Để nối kết với dữ liệu, ta cần xác định trình cung cấp OLE DB và chuỗi kết nối. Nếu không xác định được hai yếu tố này, ta sẽ sử dụng trình cung cấp mặc định là ODBC: MSDASQL.

Một số trình cung cấp có sẵn:

- Microsoft OLEDB cho các trình điều khiển ODBC.
- Microsoft OLEDB cho Oracle.
- Microsoft Jet 3.51 OLEDB (Access).
- Microsoft Jet 4.0 OLEDB (Access)
- Microsoft OLEDB cho SQL Server.
- Microsoft OLEDB cho các dịch vụ thư mục.

Ví dụ:

Đối với trình cung cấp ODBC, thuộc tính ConnectionString có thể là một DSN hay là một kết nối không có DSN (DSN cấp thấp).

```
Dim cn As ADODB.Connection

Set cn = New ADODB.Connection

cn.Provider = "MSDASQL"

cn.ConnectionString = "DSN=Baigiang"
```

cn.Open

Kết nối DSN cấp thấp:

Dim cn As ADODB.Connection

Set cn = New ADODB.Connection

cn.Provider = "MSDASQL"

cn.ConnectionString = "DRIVER={SQL Server};” & _

“DATABASE=Baigiang;UID=myuser;PWD=mypassword;"

cn.Open

Trong trường hợp này việc kết nối với cơ sở dữ liệu Server được thực hiện nhanh hơn vì chương trình không cần đọc thông tin về các DSN trên máy Client, tuy nhiên thông tin về nguồn cơ sở dữ liệu lại kết chặt với chương trình đã biên dịch.

Để kết nối với cơ sở dữ liệu Access, ta dùng trình cung cấp Jet với chuỗi kết nối là đường dẫn đến tập tin .mdb

Dim cn As ADODB.Connection

Set cn = New ADODB.Connection

cn.Provider = "MicroSoft.Jet.OLEDB.4.0"

cn.ConnectionString = "d:\data\baigiang.mdb"

cn.Open

Đối với cơ sở dữ liệu SQL Server, ta có thể dùng trình cung cấp SQLOLEDB.1, trong trường hợp này, chuỗi kết nối tương tự như trường hợp kết nối dùng trình cung cấp ODBC không có DSN, tuy nhiên ta không cần xác định giá trị của DRIVER:

```
Dim cn as ADODB.Connection

Set cn = New ADODB.Connection

cn.Provider = “SQLOLEDB.1”

cn.ConnectionString = “DATABASE=DBHH;” & _

“SERVER=www;UID=user;PWD=user”

cn.Open
```

Mở và đóng nối kết nguồn dữ liệu

Để phát các yêu cầu đến nguồn dữ liệu sử dụng ADO, ta cần mở kết nối đến nguồn dữ liệu đó bằng phương thức Open của đối tượng Connection. Cú pháp đầy đủ như sau:

```
connection.Open [connect], [userid], [password]
```

Tất cả các tham số của phương thức Open đều là tùy chọn, nếu như các thông số này đã được xác định thông qua các thuộc tính khác của đối tượng Connection thì ta không cần mô tả chúng ở đây.

Khi đã hoàn thành tất cả các thao tác liên quan đến nối kết này, ta cần phải đóng nối kết một cách tường minh thông qua phương thức Close của đối tượng Connection.

```
connection.Close
```

Đóng nối kết một cách tường minh sẽ đảm bảo rằng tất cả các tài nguyên liên quan đến nối kết này trên Server cũng như Client đều được giải phóng một cách hợp lý.

Xác định vị trí con trỏ

Con trỏ (Cursor): một tập các mẩu tin được trả về cho chương trình. Vị trí con trỏ được xác định nhờ thuộc tính CursorLocation (có ở cả đối

tượng Recordset). Có 2 giá trị có thể chỉ định:

- adUseClient: con trỏ phía Client.
- adUseServer: con trỏ phía Server (mặc định).

Thực thi các câu truy vấn hành động

Các câu truy vấn hành động (Insert, Update, Delete) được thực hiện nhờ phương thức Execute của đối tượng Connection; ngoài ra phương thức này cũng có thể được sử dụng để thực thi các thủ tục lưu trữ sẵn trong cơ sở dữ liệu hay các câu SELECT. Cú pháp phương thức này như sau:

Nếu không có kết quả trả về:

connection.Execute CommandText, RecordsAffected, Options

Có kết quả trả về:

Set recordset = connection.Execute (CommandText, RecordsAffected, Options)

Trong đó:

- connection: Đối tượng Connection.

- recordset: Đối tượng Recordset là kết quả trả về của phương thức Execute, tuy nhiên, người ta thường ít khi sử dụng cách này. Thay vào đó, người ta thường sử dụng phương thức Open của đối tượng Recordset.

- CommandText: là một chuỗi xác định câu truy vấn hành động, SELECT, thủ tục lưu trữ sẵn hay tên một bảng trong cơ sở dữ liệu.

- RecordEffected: Tùy chọn, là một số nguyên dài (Long) xác định trình cung cấp trả về bao nhiêu mẩu tin thỏa điều kiện.

- Options: Tùy chọn, là một số nguyên dài (Long) xác định trình cung cấp sẽ đánh giá các đối số của CommandText như thế nào.

Thuộc tính Mode: Xác định trình cung cấp có thể hạn chế truy cập đến cơ sở dữ liệu khi có một recordset đang mở. Các giá trị có thể là:

Hằng số	Giá trị	Ý nghĩa
adModeUnknown	0	Mặc định, chỉ định quyền hạn chưa thiết lập hay không thể xác định
adModeRead	1	Mở Recordset với quyền chỉ đọc
adModeWrite	2	Mở Recordset với quyền chỉ ghi
adModeReadWrite	3	Mở Recordset với quyền đọc/ghi
adModeShareDenyRead	4	Ngăn người khác mở kết nối với quyền chỉ đọc
adModeShareDenyWrite	8	Ngăn người khác mở kết nối với quyền chỉ ghi
adModeShareExclusive	12	Ngăn người khác mở kết nối
adModeShareDenyNone	16	Ngăn người khác mở kết nối với bất cứ quyền nào

Đối tượng Recordset

Để có thể khởi tạo một đối tượng Recordset ta có thể thực hiện một trong hai cách:

- Phương thức Execute của đối tượng Connection. Tuy nhiên cách này ta chỉ tạo được các Recordset chỉ đọc và chỉ có thể di chuyển tới.
- Xác lập các thông số thích hợp cho đối tượng Recordset rồi thực thi phương thức Open của đối tượng Recordset. Điều này được thực hiện nhờ các bước:
    - Sau khi khởi tạo đối tượng Connection, chỉ định Recordset là của đối tượng Connection trên.
    - Thiết lập các thuộc tính thích hợp của Recordset (Source, LockType…).
    - Thực thi câu truy vấn nối kết nhờ phương thức Open.

Thuộc tính CursorType (loại con trỏ)

Xác định loại con trỏ được trả về từ cơ sở dữ liệu. Các giá trị có thể nhận:

Hằng	Giá trị	Mô tả
adOpenForwardOnly	0	Chỉ có thể di chuyển phía trước
adOpenKeyset	1	Không thể thấy các mẩu tin do người dùng khác thêm vào nhưng khi họ xóa hay sửa đổi mẩu tin

		sẽ làm ảnh hưởng đến các mẩu tin ta đang làm việc.
adOpenDynamic	2	Có thể thấy toàn bộ sự thay đổi do người dùng khác tác động.
adOpenStatic	3	Bản sao tĩnh của tập mẩu tin. Mọi sự thay đổi của người dùng khác ta không thấy được

Thuộc tính LockType (khóa mẩu tin)

Xác định cách thức khóa mẩu tin trong Recordset. Dùng tính năng này khi muốn kiểm soát cách thức cập nhật mẩu tin với nhiều người dùng trong cơ sở dữ liệu.

Hằng	Giá trị	Mô tả
adLockReadOnly	1	Mặc định - Chỉ đọc.
adLockPessimistic	2	Khóa trang bi quan. Mẩu tin trong RecordSet bị khóa khi bắt đầu sửa đổi & tiếp tục khóa cho đến khi thi hành phương thức Update hay di chuyển sang mẩu tin khác.
adLockOptimistic	3	Khóa trang lạc quan. Mẩu tin chỉ bị khóa ngay lúc thi hành

		phương thức Update hay di chuyển sang mẩu tin khác.
adLockBatchOptimistic	4	Khóa trang lạc quan hàng loạt. Hỗ trợ cập nhật nhiều mẩu tin cùng một lúc.

Thuộc tính Source

Đây là một chuỗi xác định câu truy vấn để lấy dữ liệu, có thể là tên của bảng hay tên của thủ tục lưu trữ sẵn.

Thuộc tính ActiveConnection

Đây là một thuộc tính đối tượng xác định Recordset là của nối kết nào trong chương trình.

Ví dụ sử dụng đối tượng Recordset trong chương trình

Đối tượng Recordset có thể được sử dụng là đối tượng nguồn dữ liệu (DataSource) của điều khiển lưới: Microsoft DataGrid Control 6.0 (OLEDB). Nhờ điều khiển lưới này ta có thể hiển thị dữ liệu từ một Recordset theo dạng hàng và cột.

Chẳng hạn ta có thể hiển thị trên lưới thông tin về các mặt hàng cùng với mã loại hàng của nó:

```
Dim cn As ADODB.Connection

Dim rs As ADODB.Recordset

Private Sub Form_Load()
```

```
Set cn = New ADODB.Connection
cn.Provider = "Microsoft.Jet.OLEDB.3.51"
cn.ConnectionString = "F:\Data\DBHH.mdb"
cn.Open

Set rs = New ADODB.Recordset
rs.Source = "SELECT MaHang, TenHang, DVTinh” & _
“TenLoai FROM THangHoa, TLoaiHang WHERE “ & _
“THangHoa.MaLoai = TLoaiHang.MaLoai"
Set rs.ActiveConnection = cn
rs.CursorLocation = adUseClient
rs.Open
Set grdHH.DataSource = rs
End Sub
```

Kết quả thực thi của chương trình này như sau:

[missing_resource: .png]

Hình 11.4: Sử dụng Datagrid để hiển thị dữ liêu từ Recordset*

*: Microsoft DataGrid Control 6.0 (OLEDB): Name: grdHH.

Cập nhật và thêm mới mẩu tin

Thêm mới mẩu tin

-Mở Recordset

-Thi hành phương thức AddNew

-Gán giá trị cho các trường trong mẩu tin của Recordset

-Lưu lại mẩu tin bằng cách thi hành phương thức Update (hay UpdateBatch).

Cập nhật mẩu tin

-Mở Recordset

-Thực hiện câu lệnh truy vấn để nhận về các mẩu tin thích hợp.

-Di chuyển đến mẩu tin cần cập nhật lại giá trị.

-Gán lại giá trị cho các trường.

- Thi hành phương thức Update (hay UpdateBatch tùy thuộc vào LockType).

Lưu ý:Chế độ khóa mẩu tin mặc định trong ADO là chỉ đọc, vì vậy ta phải đổi thuộc tính LockType của đối tượng Recordset sang chế độ soạn thảo trước khi thi hành cập nhật hay thêm mới mẩu tin.

Thuộc tính CursorLocation

Xác định tập mẩu tin trả về từ cơ sở dữ liệu được lưu ở đâu (Server hay Client, Server là mặc định). Thuộc tính cũng giống thuộc tính CursorLocation của đối tượng Connection.

Recordset ngắt kết nối

Khi chúng ta dùng con trỏ phía Client, ta có khả năng ngắt kết nối với Server cơ sở dữ liệu mà vẫn tiếp tục làm việc với dữ liệu. Cách này cho phép ứng dụng trở nên linh hoạt hơn bởi vì nhiều người dùng có thể làm việc với cùng một dữ liệu tại một thời điểm nếu như họ không có nối kết với server.

Để ngắt nối kết với Server, ta quy định thuộc tính ActiveConnection của đối tượng Recordset là Nothing.

Ví dụ:

```
Dim cn As ADODB.Connection

Private Sub Form_Load()

Set cn = New ADODB.Connection

cn.Provider = "Microsoft.Jet.OLEDB.3.51"

cn.ConnectionString = "F:\Data\GiangDay.mdb"

cn.Open

End Sub

Public Function GetList (strState As String) _

As ADODB.Recordset

Dim rs As ADODB.Recordset

Set rs = New ADODB.Recordset

Set rs.ActiveConnection = cn

rs.CursorLocation = adUseClient

rs.LockType = adLockBatchOptimistic
```

```
rs.CursorType = adOpenKeyset

rs.Open strState

Set rs.ActiveConnection = Nothing

Set GetList = rs

Set rs = Nothing

End Function
```

Để thi hành cùng một hành động trên một mẩu tin, ta sửa đổi lại các thuộc tính của đối tượng Recordset.

```
rs.LockType = adLockBatchOptimistic

rs.CursorType = adOpenKeyset
```

Chúng ta thiết lập giá trị các thuộc tính lại như trên để xác nhận rằng Recordset có thể nối kết lại để cập nhật về sau.

Sau đó, ta sẽ thiết lập một hàm nhận Recordset ngắt kết nối làm tham biến để tạo một đối tượng Recordset khác cập nhật dữ liệu.

```
Public Sub WriteData(rsDis As ADODB.Recordset)

Dim rs As ADODB.Recordset

Set rs = New ADODB.Recordset

Set rs.ActiveConnection = cn

rs.Open rsDis, cn

rs.UpdateBatch

End Sub
```

Gọi thực thi thủ tục WriteData:

```
Private Sub cmdWrite_Click()

WriteData GetList("Select * From THanghoa")

End Sub
```

Đối tượng Command

Đây là đối tượng được người lập trình sử dụng khi muốn thi hành các thủ tục lưu trữ sẵn hay những câu truy vấn có tham số.

Với đối tượng Command ta có thể thi hành một số công việc như sau:

- Sử dụng thuộc tính CommandText để định nghĩa các đoạn Text thi hành được. Thông thường thuộc tính này dùng để thiết lập một câu lệnh SQL hoặc một lời gọi thủ tục lưu trữ sẵn, hay những dạng khác mà trình cung cấp hỗ trợ

- Xây dựng chuỗi các đối số của câu truy vấn cũng như các tham số của các thủ tục lưu trữ sẵn thông qua đối tượng Parameter hoặc tập hợp Parameters.

- Thực hiện một câu truy vấn và trả về đối tượng Recordset thông qua phương thức Execute.

- Xác định kiểu của đối tượng Command để nâng cao hiệu quả thông qua thuộc tính CommandType.

- Xác định số giây mà trình cung cấp phải chờ khi thi hành một đối tượng Command thông qua thuộc tính CommandTimeOut.

Các kiểu của đối tượng Command được trình bày trong bảng dưới đây:

Hằng	Ý nghĩa
adCmdText	Định giá thuộc tính CommandText dưới dạng Text của một câu lệnh hoặc một lời gọi thủ tục lưu trữ sẵn.
adCmdTable	Định giá thuộc tính CommandText như là tên của một bảng khi tất cả các trường của bảng đó sẽ được trả về bởi câu lệnh truy vấn nội tại.
adCmdTableDirect	Định giá thuộc tính CommandText như là tên của một bảng khi mà tất cả các trường của bảng đó sẽ được trả về.
adCmdStoredProc	Định giá thuộc tính CommandText như là tên của một thủ tục lưu trữ sẵn.
adExecuteNoRecords	Chỉ định rằng thuộc tính CommandText là một câu lệnh hoặc một thủ tục lưu trữ sẵn không trả về bất kỳ dòng nào (ví dụ như lệnh thêm mới dữ liệu …). Cấu trúc này luôn bao hàm adCmdText, adCmdStoredProc.

Thuộc tính Parameter được xác lập thông qua hai phương thức CreateParameter và Append

Set parameter = command.CreateParameter (Name, Type, _

Direction, Size, Value)

- Name: tùy chọn, chuỗi xác định tên của đối tượng Parameter.
- Type, Direction: giá trị xác đin kiểu của đối tượng Parameter
- Size: giá trị xác định độ dài tối đa của giá trị đối tượng Parameter bằng ký tự hoặc Byte.

- Value: biến xác định giá trị của Parameter truyền.

Những giá trị có thể của thuộc tính Direction:

Hằng	Mô tả
adParamUnknown	Không biết chiều của Parameter.
adParamInput	Mặc định, xác định đây là tham số đầu vào.
adParamOutput	Tham số đầu ra.
adParamInputOutput	Vừa là tham số đầu vào vừa là tham số đầu ra.
adParamReturnValue	Đây là giá trị trả về.

Phương thức Append dùng để đưa đối tượng Parameter vừa tạo vào tập hợp. Chúng ta sẽ xét qua ví dụ dau đây:

```
Public Sub ActiveConnectionX()

Dim cnn1 As ADODB.Connection

Dim cmdByRoyalty As ADODB.Command

Dim prmByRoyalty As ADODB.Parameter

Dim rstByRoyalty As ADODB.Recordset

Dim rstAuthors As ADODB.Recordset
```

```
Dim intRoyalty As Integer
Dim strAuthorID As String
Dim strCnn As String
' Định nghĩa 1 đối tượng command cho một thủ tục lưu trữ sẵn
Set cnn1 = New ADODB.Connection
cnn1.Provider = "SQLOLEDB.1"
cnn1.ConnectionString = "DATABASE=Pubs;" & _
"SERVER=(local);UID=user;PWD=user;"
cnn1.Open
Set cmdByRoyalty = New ADODB.Command
Set cmdByRoyalty.ActiveConnection = cnn1
cmdByRoyalty.CommandText = "byroyalty"
cmdByRoyalty.CommandType = adCmdStoredProc
cmdByRoyalty.CommandTimeout = 15
' Định nghĩa đối số đầu vào cho thủ tục lưu trữ
intRoyalty = Trim(InputBox( "Enter royalty:"))
Set prmByRoyalty = New ADODB.Parameter
prmByRoyalty.Type = adInteger
prmByRoyalty.Size = 3
prmByRoyalty.Direction = adParamInput
```

```
prmByRoyalty.Value = intRoyalty
cmdByRoyalty.Parameters.Append prmByRoyalty
' Tạo một recordset bằng cách thi hành đối tượng Command.
Set rstByRoyalty = cmdByRoyalty.Execute()
' Mở bảng Authors để lấy tên hiển thị
Set rstAuthors = New ADODB.Recordset
rstAuthors.Open "authors", cnn1, , , adCmdTable
Debug.Print "Authors with " & intRoyalty & _
" percent royalty"
Do While Not rstByRoyalty.EOF
strAuthorID = rstByRoyalty!au_id
Debug.Print , rstByRoyalty!au_id & ", ";
rstAuthors.Filter = "au_id = '" & _
strAuthorID & "'"
Debug.Print rstAuthors!au_fname & " - " & _
rstAuthors!au_lname
rstByRoyalty.MoveNext
Loop
rstByRoyalty.Close
rstAuthors.Close
```

```
cnn1.Close

End Sub
```

Đối tượng Field

Dùng đối tượng Field và tập hợp Fields khi ta muốn truy cập giá trị của một trường của một Recordset nào đó, kỹ thuật này tương tự như đối với DAO.

Dịch vụ dữ liệu từ xa của ADO

Đây là kỹ thuật sử dụng thư viện Remote Data Service (RDS) để vận chuyển ADO Recordset từ server đến máy tính client Recordset kết quả được lưu ở máy client và chúng được ngắt kết nối đến server.

RDS là một phần của Microsoft Data Access Components (MDAC). Các thông tin về RDS có thể tìm thấy ở trang http://www.microsoft.com/data/. RDS gồm 2 phần chính:

- RDS 1.5 server: đi kèm khi cài đặt Internet Information Server (IIS) 4.0.

- RDS 1.5 client đi kèm khi cài đặt Internet Explorer (IE) 4.0.

Thư viện ADODB gồm các thành phần hoạt động chủ yếu phía server (server side) như các đối tượng Connection, Command, Error, Parameters ... Sẽ thật hiệu quả nếu sử dụng các thành phần này giao tiếp với cơ sở dữ liệu. Tuy nhiên trong trường hợp sử dụng các chức năng cần phải có ở phía client thì ta cần phải phân phối kèm theo một số tập tin và sử dụng ODBC cho mỗi máy client. Đối tượng ADODB Recordset không thể phân phối với các thành phần của RDS Client. Thay vào đó thư viện đối tượng Microsoft ActiveX Data Objects RecordSet (ADOR) được sử dụng. Thư viện này gồm các thành phần hoàn toàn nằm ở phía client và cho phép ta có các thao tác trên một recordset thật sự phía client. ADOR không có các

đối tượng Connection, Command, Error, hay Parameters. ADOR có các chức năng cho phép phân phối recordset với các thành phần RDS client.

Một ADO Recordset không thể vận chuyển thông qua giao thức http. Thay vào đó RDS được sử dụng để nhận và tương tác dữ liệu từ xa thông qua http. Một proxy RDS được sử dụng để kiểm soát từ xa một ADOR Recordset ngắt kết nối truyền thông qua giao thức http. Như vậy RDS là vật chứa (container) cho phép lưu trữ và truy cập từ xa các ADOR Recordset.

Ta có thể dùng đối tượng DataControl của RDS để nhận về đối tượng Recordset của ADO từ Internet.

Để có thể sử dụng kỹ thuật này, ta cần tham khảo các thuộc tính chủ yếu của đối tượng DataControl.

- Thuộc tính Connect:

DataControl.Connect = "DSN=DSNName;UID=usr;PWD=pw;"

- Thuộc tính Server: Xác định máy chủ Web chứa nguồn dữ liệu bao gồm tên và giao thức nối kết.

- Thuộc tính SQL: Là câu lệnh truy vấn để nhận về đối tượng Recordset

DataControl.SQL = "QueryString"

- Thuộc tính ExecuteOptions: xác định việc thi hành các câu lệnh truy vấn một cách đồng bộ hay không, các giá trị là một trong hai giá trị sau đây:

Hằng	Mô tả
adcExecSync	Thi hành đồng bộ

adcExecAsync	Mặc định, Thi hành không đồng bộ.

- Thuộc tính ReadyState: Xác định trạng thái của điều khiển.

Giá trị	Mô tả
adcReadyStateLoaded	Câu truy vấn hiện hành vẫn đang còn thực hiện và chưa có một dòng nào được trả về. Đối tượng Recordset của RDS.DataControl chưa thể sử dụng.
adcReadyStateInteractive	Tập hợp dòng ban đầu đã được trả về và chứa trong đối tượng Recordset, các dòng tiếp theo vẫn đang được trả về.
adcReadyStateComplete	Tất cả các dòng đều đã được chứa trong đối tượng Recordset.

- Phương thức Refresh: thi hành câu truy vấn.

- Thuộc tính Recordset: trả về Recordset kết quả.

Recordset = DataControl.Recordset

- Phương thức DoEvents: Đây là hàm của VB, nó sẽ trả điều khiển cho hệ điều hành thực hiện các quá trình khác.

Môi trường dữ liệu
Mục tiêu: Chương này gồm các bài tập nhằm rèn luyện cho sinh viên cách thức sử dụng môi trường dữ liệu (Data Environment) của VB để lập trình CSDL.

Học xong chương này, sinh viên phải nắm bắt được các vấn đề sau:

Sử dụng thành thạo môi trường dữ liệu gồm:

- Tạo đối tượng Connection.

- Tạo đối tượng Command.

- Viết mã lệnh thao tác với môi trường dữ liệu.

Kiến thức có liên quan:

- Giáo trình Visual Basic, Chương 12.

Tài liệu tham khảo:

- Visual Basic 6 Certification Exam Guide – Chapter 10, Page 277 - Dan Mezick & Scot Hillier - McGraw-Hill - 1998.

- Tự học Lập trình cơ sở dữ liệu với Visual Basic 6.0 trong 21 ngày (T1) - Chương 9, trang 395 - Nguyễn Đình Tê (chủ biên) – Nhà xuất bản Giáo dục - 2001.

HƯỚNG DẪN

Bài 7-1

DATA ENVIRONMENT

Data Environment cho phép chúng ta tạo ứng dụng cơ sở dữ liệu với OLEDB một cách nhanh chóng và hiệu quả. Trong bài tập này, ta sẽ tìm hiểu về Data Environment và Report Designer của VB.

Bước 1: Tạo thư mục Basic\Bt7-1. Tạo một dự án kiểu Stardard EXE lưu vào trong thư mục đó.

Bước 2: Nếu mục Data Environment không có sẵn trong Project Explorer, ta chọn Project\Components…, đánh dấu vào mục Data Environment trong tùy chọn Designers, nhấp OK. Chọn Project\More ActiveX Designers… để thêm Data Environment vào môi trường soạn thảo.

Bước 3: Trong Data Environment, nhấp chuột phải vào đối tượng Connection1, chọn Properties…, chọn Microsoft Jet 3.51 OLE DB Provider.

Bước 4: Trong mục chọn Connection, chọn cơ sở dữ liệu mình muốn thao tác trong mục Select or Enter a Database Name Box; ở đây ta chọn CSDL BIBLIO.MDB (thường ở đường dẫn C:\Program Files\Microsoft Visual Studio\VB98\BIBLIO.MDB). Nhấp nút Test Connection để kiểm tra nối kết với CSDL có bị lỗi hay không? Ta sẽ nhấp OK nếu nối kết này thành công (nếu không ta phải kiểm tra lại).

Bước 5: Trong Data Environment, nhấp chuột phải vào đối tượng Connection1 và chọn RENAME để đổi tên thành BIBLIO.

Bước 6: Nhấp chuột phải vào BIBLIO và chọn ADD COMMAND trên menu, một đối tượng command được tạo ra với tên là Command1 trong Data Environment. Nhấp chuột phải vào đối tượng mới tạo này, chọn RENAME để đổi tên thành Publishers.

Bước 7: Nhấp chuột phải vào Publishers và chọn mục Properties, một hộp thoại quy định các thuộc tính cho đối tượng Publishers được mở ra. Trong mục chọn General, chọn Source Data là SQL Statement và ta nhập câu SQL sau vào khung nhập:

SELECT PubID, Name FROM Publishers WHERE Name LIKE ?

Câu SQL trên phải nhận vào một tham số từ chương trình gọi nó (dấu chấm hỏi). Nghĩa là để câu SQL này thực thi được, ta cần cung cấp một tham số đầu vào cho nó.

Bước 8: Để định nghĩa tham số, ta chuyển sang mục chọn Parameters, ta thấy có một tham số đã được định nghĩa tên là Param1. Mặc nhiên của tham số này là kiểu số, tuy nhiên ta sẽ đổi chúng thành kiểu chuỗi với các thuộc tính như sau:

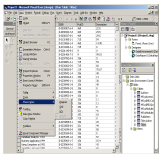
Data TypeadVarChar

Host Data TypeString

Size255

Bước 9: Tạo một recordset con cho đối tượng Publishers command bằng cách nhấp chuột phải vào vào đối tượng này và chọn ADD CHILD COMMAND trên menu. Các câu truy vấn này dùng để tìm thông tin về các sách dựa vào nhà xuất bản nào đó. Khi một command con được thêm vào, nó có tên là COMMAND1.

Bước 10: Trong command con này, nó cần phải có một câu SQL để truy xuất thông tin. Câu truy vấn này có nhiệm vụ kết nối thông tin từ các bảng Title và Author đối với từng loại nhà xuất bản.



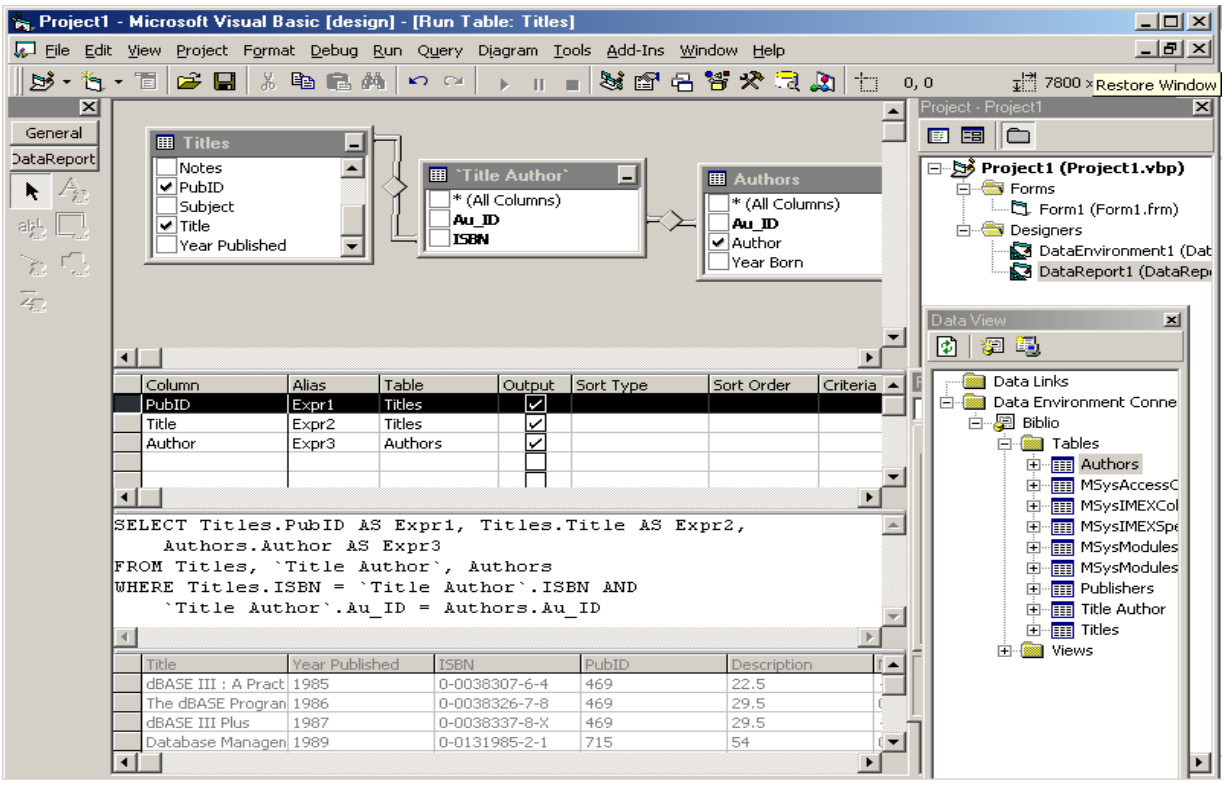
Bước 11: Mở cửa sổ Data View bằng cách chọn View\Data View Window trên menu. Trong Data View, xác định thư mục Data Environment Connections. Đối tượng connection BIBLIO được hiển thị trong thư mục này. Mở đối tượng connection và xác định thư mục Tables. Trong thư mục này, chọn bảng Titles. Nhấp đúp vào bảng Titles này để hiển thị nội dung của bảng.

Bước 12: Khi đã hiển thị nội dung của bảng, ta mở Data Tools bằng cách chọn VIEW\SHOW PANES trên menu. Chọn tất cả các mục trong phần này (Diagram, Grid, SQL, Results).

Bước 13: Khi tất cả các mục của Data Tool được chọn, ta kéo các bảng Title Author và Authors vào khung diagram của Data Tools. Sau đó ta phải kiểm tra các liên kết của các bảng: bảng Titles có liên kết với bảng Title Author bởi trường ISBN không? Bảng Title Author có liên kết với bảng Authors bởi trường Au_ID không? Nếu không có các mối liên kết này ta phải chọn đúng trường trong bảng Title Author và kéo chúng vào trường tương ứng trong 2 bảng Titles và Authors.

Bước 14: Để tạo ra câu truy vấn, ta đánh dấu chọn vào trường PubID, Title trong bảng Titles; trường Author trong bảng Authors. Kế tiếp, kiểm tra khung Grid ta thấy có một dòng có dấu * (như hình dưới) biểu thị là ta đã chọn tất cả các trường của tất cả các bảng. Ta không cần chọn tất cả thông tin, do đó ta nhấp vào dòng có dấu * và bấm phím Delete để xóa chúng. Lúc đó trong khung SQL ta thấy có câu SQL sau:

```
SELECT Titles.PubID AS Expr1, Titles.Title AS Expr2,

Authors.Author AS Expr3

FROM Titles, `Title Author`, Authors

WHERE Titles.ISBN = `Title Author`.ISBN AND

`Title Author`.Au_ID = Authors.Au_ID
```



Hình VII.1: SQL Builder

Bước 15: Chọn câu SQL hiển thị bên trên rồi nhấp Edit\Copy.

Bước 16: Nhấp đúp trở lại vào Data Environment, nhấp chuột phải vào command con đã có và chọn PROPERTIES trên menu. Sau đó đổi tên của command trong mục General thành Titles. Chọn Source Data là SQL Command; sau đó dán nội dung câu SQL đã copy ở trên vào khung nhập câu SQL (chọn Edit\Paste).

Bước 17: Chọn mục RELATION , ta thấy trường PubID của cả hai bảng Publishers và Titles được hiển thị. Nhấp nút ADD để xác định liên kết giữa câu command cha và command con.

Bước 18: Đóng các cửa sổ và lưu dự án lại.

XÂY DỰNG GIAO DIỆN CHƯƠNG TRÌNH

Bước 19: Thêm vào một điều khiển vào dự án bằng cách chọn Project\Components trên menu; tìm đến mục Microsoft Hierachial Flexgrid Control 6.0 (OLEDB). Đánh dấu tùy chọn này và nhấp OK.

Bước 20: Mở Form1, tạo giao diện cho chương trình như dạng sau (hình bên dưới):

Item 1: Label

Name: lblCompany

Caption: Company Name

Item 2: TextBox

Name: txtCompany

Text: micro

Item 3: CommandButton

Name: cmdGO

Caption: GO

Default: TRUE

Item 4: Hierarchial FlexGrid

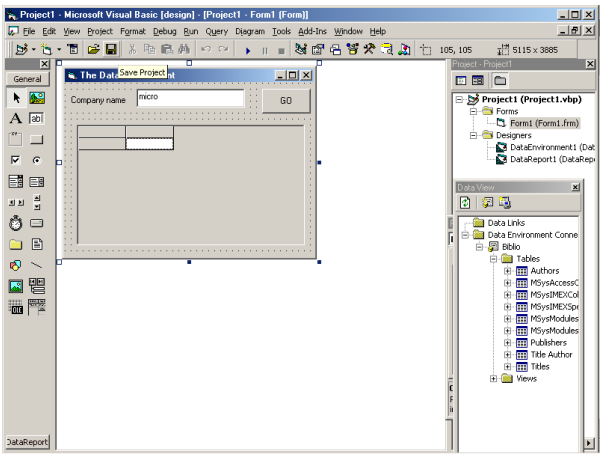
Name: grdTitles

AllowUserResizing: 1-flexResizeColumns

DataSource: DataEnvironment1

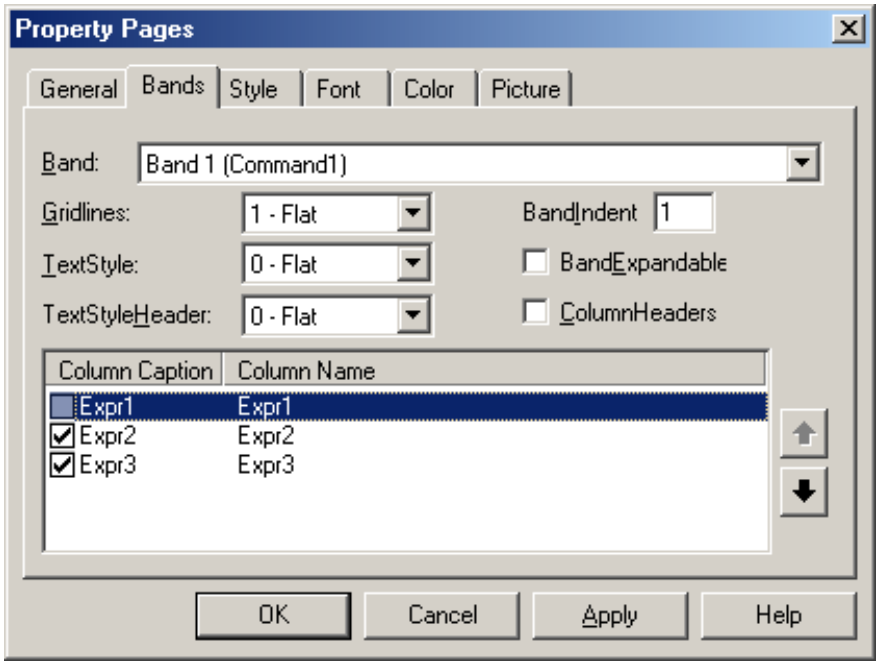
DataMember: Publishers

FixedCols: 0



Hình VII.2: Giao diện ứng dụng

Bước 21: Nhấp chuột vào phải vào Hierarachical FlexGrid trên Form1, hộp thoại thuộc tính của Hierarachical được mở ra. Chọn mục BANDS trong hộp thoại này. Chọn Band0 (Publishers) sau đó không đánh dấu vào trường PubID (với tên là Expr1) để nó không hiển thị khi thực hiện câu SQL. Cũng vậy, không đánh dấu vào trường PubID của Band1 (hình dưới):



Hình VII.3: Chọn trường hiển thị trên lưới

Bước 22: Đổi tên của các cột trong Band1 từ Expr2, Expr3 thành Title và Author bằng cách: Nhấp chuột 2 lần vào mục cần đổi tên trong Column Caption, nhập tên mới vào; sau đó chọn OK.

Bước 23: Mục đích của chương trình này là: Khi chương trình thực thi, trong TextBox có một từ là micro, từ được đề nghị tìm kiếm. Khi nhấp chuột vào nút nhấn; đoạn văn bản trong TextBox được dùng để thực thi câu SQL Publishers; câu SQL này sẽ truy tìm tất cả các công ty mà tên của chúng chứa chuỗi được nhập vào trong TextBox; kết quả trả về được hiển thị trên lưới. Do đó, ta sẽ xử lý sự kiện cmdGO_Click bằng đoạn mã sau:

```
MousePointer = vbHourglass

With DataEnvironment1

.rsPublishers.Close

.Publishers "%" & txtCompany.Text & "%"

End With

' Thiet lap tren luoi

Set grdTitles.DataSource = DataEnvironment1

grdTitles.CollapseAll

MousePointer = vbDefault
```

Bước 24: Lưu dự án và chạy chương trình. Sử dụng micro để tìm kiếm (ta có thể thử với từ khác như: hill hay mill)

TẠO BÁO CÁO (REPORT)

Bước 25: Nếu Data Report Designer không hiển thị dưới menu PROJECT, mở hộp thoại COMPONENTS, chọn Data Report Designer trong mục chọn Designers. Sau đó ta thêm Data Report Designer vào dự án bằng cách chọn PROJECT\ADD DATA REPORT trên menu.

Bước 26: Liên kết Data Report Designer với Data Environment nhờ việc thiết lập thuộc tính DataSource của Data Report là DataEnvirronment1 và thuộc tính DataMember là Publishers.

Bước 27: Nhấp chuột phải trên Data Report Designer và chọn RETRIEVE STRUCTURE trên menu. Một hộp thoại xác nhận hiện lên và ta chọn YES.

Bước 28: Ta nhận thấy trên report gồm các phần: report header, page header, publisher information, title information, page footer, và report footer.

Bước 29: Chọn phần Report Header, trong phần này, thêm một Report Label vào, đặt Caption là Book Report; ta có thể đổi Font chữ và kích thước Font tương ứng.

Bước 30: Chọn Group header, trong phần này thêm một Report TextBox vào; đặt thuộc tính DataMember là Publishers và DataField là Name.

Bước 31: Chọn phần Detail của Report. Trong phần này, thêm 2 Report TextBox vào, đặt 2 điều khiển này cạnh nhau trong phần Detail; chọn TextBox thứ nhất và đặt thuộc tính DataMember là Titles và DataField là Expr2, đối với TextBox thứ hai: DataMember: Titles, DataField: Expr3.

Bước 32: Lưu dự án lại.

Bước 33: Chọn Form1 trong môi trường soạn thảo; tạo một menu trong Form1 bằng cách chọn Tools\Menu Editor; trong Menu Editor, tạo một Report menu với các phần tử là Preview và Print với các thuộc tính sau:

Item 1 – Menu

Name: mnuReport

Caption: Report

Item 2 – Menu

Name: mnuPreview

Caption: Preview

Item 3 – Menu

Name: mnuPrint

Caption: Print

Bước 34: Ta có thể xem báo cáo trước khi in nhờ hàm Show của đối tượng Report. Trong hàm xử lý sự kiện mnuPreview, thêm đoạn mã sau:

DataReport1.Show

Bước 35: In báo cáo được thực hiện nhờ hàm PrintReport của đối tượng Report. Trong hàm xử lý sự kiện mnuPrint, thêm đoạn mã sau:

DataReport1.PrintReport

Bước 36: Lưu và chạy chương trình. Thử hiển thị và in report.

Bài 7-2

BIỂU MẪU NHẬP LIỆU VỚI DATA ENVIRONMENT

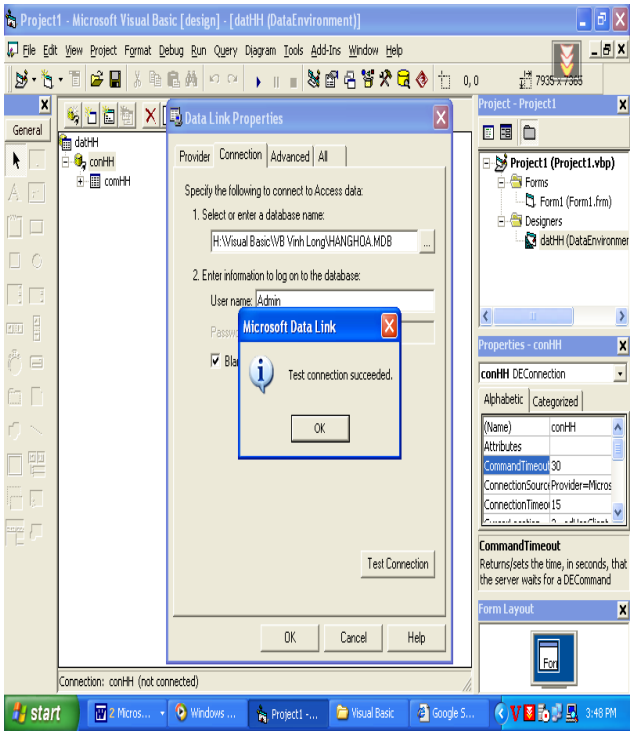
- Tạo một dự án mới; bổ sung Data Enviroment vào dự án của ta nhờ chọn Project/Add Data Environment. Khi lựa chọn mục này, môi trường DED sẽ hiển thị; sử dụng cửa sổ Properties để thiết lập thuộc tính Name là: datHH. Ở đây, ta sẽ sử dụng DED để kết nối với CSDL HANGHOA.MDB.

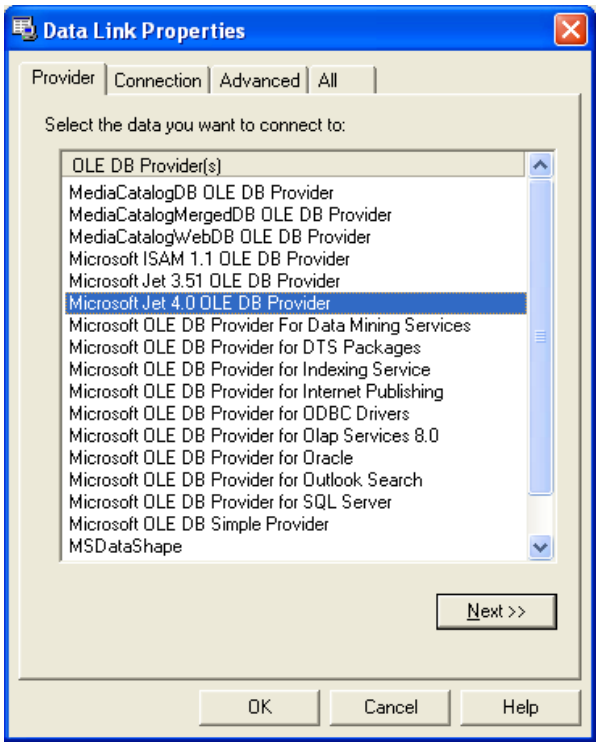
- Sửa lại thuộc tính Name của Connection1 là conHH; sau đó chuột phải lên conHH, chọn Properties.

Ở hộp thoại đầu tiên, ta phải chọn một trình cung cấp dữ liệu, ở đây chọn Microsoft Jet 4.0 OLE DB Provider, nhấn Next để tiếp tục.

Tiếp theo ta cần nhập chính xác đường dẫn đến tập tin CSDL, chẳng hạn ở đây là: H:\Visual Basic\HangHoa.Mdb.

Cuối cùng, nhấn nút Test Connection để kiểm tra việc nối kết dữ liệu chính xác hay không?



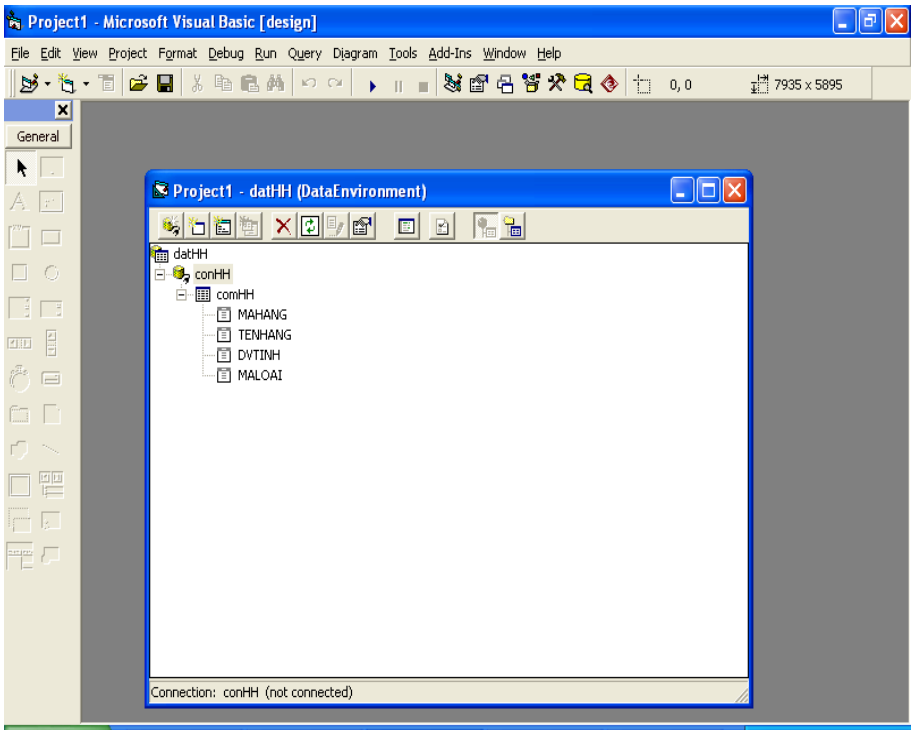


Hình VII.4: Tạo Connection

Tạo đối tượng Command:

- Xây dựng một đối tượng Command kết nối trực tiếp với Table (bảng) THANGHOA trong file dữ liệu HangHoa.mdb.
- Nhấp chuột phải trên kết nối dữ liệu conHH & chọn Add Command; sửa Command Name là: comHH; chọn Table từ Combo Box Database Object, chọn THANGHOA từ Combo Box Object Name.
- Trước khi đóng hộp thoại này, ta chuyển qua nhãn Advanced & thiết lập LockType là 3 – Optimistic (mặc nhiên là 1 – Read Only); Cursor Location: Use client-side cursor. Nhờ vậy ta mới có thể cập nhật Record Set từ chương trình của ta.
- Trở lại giao diện DED, ta được:

Hình VII.5: Đối tượng Command



Tạo một ứng dụng nhập liệu với DED

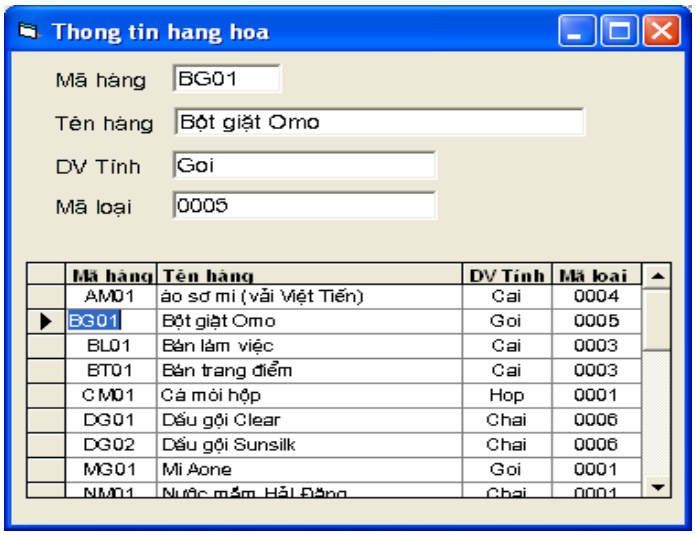
- Ở môi trường DED, ta kéo các trường của Command comHH vào Form1, chỉnh sửa lại cho thích hợp.
- Ở đây ta có sử dụng một lưới để hiển thị dữ liệu; do vậy ta chọn Project\Component; chọn Microsoft DataGrid Control 6.0 (OLE DB); sau đó kéo điều khiển này vào Form, thiết lập các thuộc tính cho thích hợp.

Name: grdHH.

DataSource: datHH

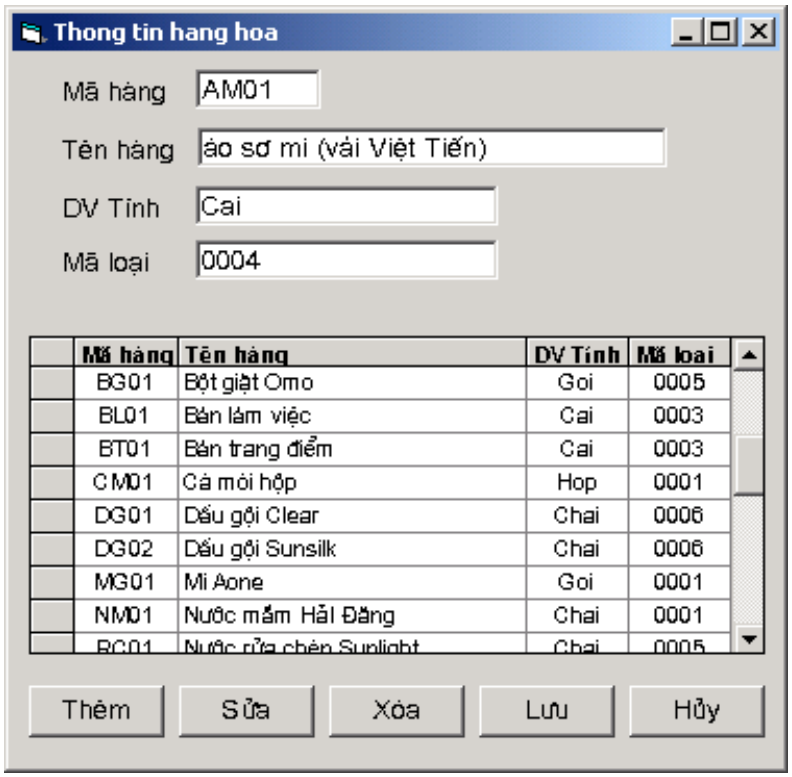
DataMember: comHH

- Nhấp chuột phải lên điều khiển DataGrid, chọn Retrieve Structure. Sau đó, lưu dự án & chạy chương trình ta được:



Hình VII.6: Kết quả thực thi ứng dụng

- Thêm các nút hành động (Thêm, Sửa, Xóa,…). Chẳng hạn các sự kiện cmd_Them_Click, cmdXoa_Click, cmdLuu_Click, cmdHuy_Click được xử lý:



Hình VII.7: Giao diện đầy đủ

```
Private Sub cmdThem_Click()
With datHH.rscomHH
.AddNew
End With
End Sub
Private Sub cmdXoa_Click()
With datHH.rscomHH
.Delete
.Update
Me.Refresh
End With
End Sub
Private Sub cmdHuy_Click()
With datHH.rscomHH
.CancelUpdate
Me.Refresh
End With
End Sub
Private Sub cmdLuu_Click()
```

On Error GoTo Xuly

With datHH.rscomHH

.Update

End With

Me.Refresh

Exit Sub

Xuly:

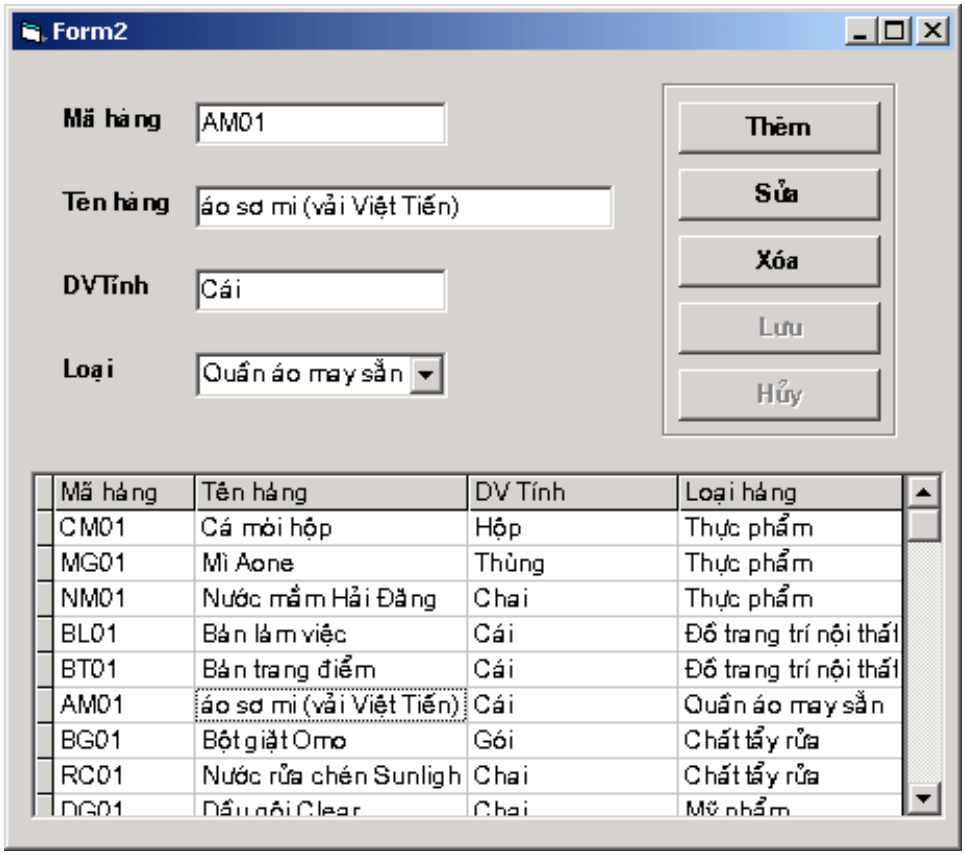
MsgBox Err.Description, vbCritical + vbSystemModal, "Error"

End Sub

Như vậy, ta đã thiết kế xong một Form cho phép hiển thị thông tin các hàng hóa, Form này cho phép sửa đổi, thêm mới các mẩu tin trong bảng THANGHOA của CSDL HANGHOA.MDB.

BÀI TẬP TỰ LÀM

1) Sử dụng DataEnviroment, thiết kế Form nhập liệu cho bảng THangHoa (hình dưới). Ở đây thay vì hiển thị MaLoai, ta lại hiển thị TenLoai:



Hình VII.8: Form nhập liệu

2) Sử dụng DataEnviroment, thiết kế Form nhập liệu cho bảng TNhanVien.

3) Sử dụng DataEnviroment, thiết kế Form cho phép nhập (sửa, xóa) thông tin về một phát sinh về một mặt hàng nào đó trong ngày. Lưu ý: Trường STT là kiểu AutoNumber (Access), Ngay: lấy ngày hệ thống (hàm Now).

Thiết lập báo cáo
Mục tiêu: Chương này giới thiệu cách thức để tạo báo cáo bao gồm hiển thị dữ liệu cũng như sắp xếp và phân nhóm dữ liệu.

Học xong chương này, sinh viên có thể:

- Sử dụng tính năng Report của Microsoft Access trong các ứng dụng nhỏ.
- Sử dụng Data Report để tạo báo biểu.
- Sử dụng Crystal Report, công cụ mạnh để tạo báo biểu.

Kiến thức cần thiết:

- Thư viện đối tượng ActiveX Data Objects (ADO).
- Môi trường dữ liệu Data Environment.

Tài liệu tham khảo:

Visual Basic 6.0 và Lập trình cơ sở dữ liệu - Chương 21, trang 637 - Nguyễn Thị Ngọc Mai (chủ biên) – Nhà xuất bản Giáo dục - 2001.

SỬ DỤNG MICROSOFT ACCESS ĐỂ LẬP BÁO CÁO

Có hai kỹ thuật để thi hành một báo cáo Access từ ứng dụng VB:

- Sử dụng Automation để phóng một thể hiện (instance) của Microsoft Access, thi hành báo cáo trực tiếp từ trong ứng dụng. Automation là một kỹ thuật cho phép giao tiếp giữa các ứng dụng trên Windows. Ở đây Microsoft Access sẽ làm Automation Server.

- Dùng VSREPORTS của VideoSoft cho phép người sử dụng VB thi hành báo cáo của Microsoft Access bất kể máy của họ có cài đặt Microsoft Access hay là không. Đây là một điều khiển ActiveX chuyển đổi báo cáo từ tập tin MDB thành một định dạng mà ta có thể cung cấp cùng ứng dụng.

Trong bài giảng này, chúng tôi chỉ trình bày cách thứ nhất mặc dù cách này có nhiều hạn chế. Đối với cách thứ hai, để có thể thực hiện được ta cần phải cài đặt một số thư viện liên kết động (DLL). Các thư viện này tương đối khó tìm và nhất là chúng đòi hỏi bản quyền.

Bất lợi của kỹ thuật dùng Automation là buộc người dùng phải chạy một thể hiện (instance) của Microsoft Access cũng như phải cài đặt Microsoft Access trên máy. Để lập trình theo kỹ thuật này, ta tiến hành theo các bước sau:

- Tham chiếu đến Microsoft Access bằng cách từ menu Project chọn Preferences -> Microsoft Access 9.0 Object Library.

- Sau đó ta tạo một đối tượng như là đối tượng ứng dụng của Access như sau:

Dim MSAccess As Access.Application

- Sau đó ta cần tạo mới đối tượng này cũng như tạo một tham chiếu đến cơ sở dữ liệu chứa báo cáo:

Set MSAccess = New Access.Application

MSAccess.OpenCurrentDatabase(“Database Name”)

- Sử dụng thuộc tính DoCmd để thi hành báo cáo:

MSAccess.DoCmd.OpenReport "Report Name",acViewNormal

- Đóng cơ sở dữ liệu:

MSAccess.CloseCurrentDatabase

Lưu ý: Tránh dùng ràng buộc trễ với Automation

Phiên bản cũ của Automation là OLE Automation, dùng trong VB 3.0 và Microsoft Access 2.0.

Trong VB 3.0, ta có thể viết chương trình như sau:

Dim MSAccess As Object

Set MSAccess = CreateObject(“Access.Application”)

Đoạn chương trình trên hoạt động tốt đối với VB 3.0 nhưng có một cách khác tốt hơn. Thay vì dùng kiểu Object, ta nên chỉ rõ kiểu dữ liệu đối tượng mà Automation Server cung cấp (chẳng hạn Access.Aplication nếu là Access). Bởi vì khi đó, VB không cần thi hành câu truy vấn trên Automation Server mỗi khi ta truy cập nó để xác định kiểu đối tượng cần tạo. Kỹ thuật này gọi là ràng buộc trễ, giờ đây chỉ phủ hợp với 2 tình huống:

- Ta không biết trước kiểu đối tượng Automation Server.
- Ta đang sử dụng một môi trường phát triển ứng dụng không hỗ trợ ràng buộc sớm, như VBScript hay ASP.

SỬ DỤNG THIẾT KẾ DATA REPORT

Thiết kế báo cáo dùng DataReport là điểm mới trong VB6, đây là một công cụ được hỗ trợ bởi VB6, cung cấp một cách trực quan về thiết kế báo cáo và có ưu diểm là rất dễ dùng.

Thiết kế với DataReport

- Chọn Project -> Components.

- Chọn Tab Designers, đánh dấu chọn Data Report.

[missing_resource: .png]

Hình 13.1 Đưa thiết kế báo cáo về đề án

Các thành phần của một báo cáo như sau:

- Report Header: Hiển thị một lần ở đầu báo cáo.

- Report Footer: Hiển thị một lần ở cuối báo cáo.

- Page Header: Hiển thị tại đầu mỗi trang.

- Page Footer: Hiển thị tại cuối mỗi trang.

- Detail Section: Hiển thị các dòng dữ liệu.

- Một hoặc nhiều nhóm đầu cuối hiển thị tại đầu và cuối mỗi phân nhóm.

Các điều khiển của thiết kế Data Report như sau:

- Điều khiển nhãn (Rpt Label).

- Điều khiển hộp văn bản (Rpt Textbox).

- Điều khiển ảnh (Rpt Image).

- Điều khiển hình dạng (Rpt Shape).

- Điều khiển các hàm tính toán (Report Function: rptFuncSum, rptFuncAve, rptFuncMin, rptFuncMax…).

[missing_resource: .png]

Hình 13.2 Cửa sổ Data Report

Các điều khiển của Data Report cũng giống như là các điều khiển chuẩn trên biểu mẫu, chúng có thể ràng buộc với nguồn dữ liệu. Tuy nhiên, ta có một cách thức khác dễ dàng hơn đó là sử dụng môi trường dữ liệu (được giới thiệu ở chương trước).

Sử dụng DataEnvironment trong việc tạo DataReport:

Quá trình thực hiện trải qua các bước sau:

- Tạo đối tượng Command.

- Kéo thả các trường của đối tượng Command này vào thiết kế của Report.

- Thêm các tiêu đề đầu trang & cuối trang.

Ví dụ: Tạo báo cáo về các sinh viên trong bảng STUDENT thuộc cơ sở dữ liệu Student.

[missing_resource: .png]

Hình 13.3 Thiết lập Data Environment- Bước 1: Tạo một nối kết đến CSDL Student trong trình Data Environment, thêm một đối tượng Command cho phép lấy dữ liệu từ bảng Student.

- Bước 2: Kéo thả các trường cần hiển thị vào báo cáo tại mục Detail, chỉ giữ lại trường liên quan đến thông tin dữ liệu (đặt trong phần Detail Section). Thiết lập tên trường dưới dạng tiếng Việt tại phần Page Header.

- Bước 3: Cung cấp các thông tin cho phép DataReport nhận dữ liệu từ đâu bằng cách xác lập: DataSource: DataEnvironment1, DataMember: Student.

[missing_resource: .png]

Hình 13.4: Report khi đã kéo thả các trường

Thiết kế báo biểu có phân nhóm dữ liệu

[missing_resource: .png]

Hình 13.4: Nhóm dữ liệu- Chọn đối tượng Command của trình DataEnvironment cần nhóm cơ sở dữ liệu.

- Hiển thị trang thuộc tính, chọn Tab Grouping.

- Chọn tùy chọn Group Command Object.

- Đặt tên cho nhóm cũng như chọn các trường tham gia vào nhóm dữ liệu.

- Đặt lại giá trị cho thuộc tính Data Member chỉ đến nối kết mới đã nhóm dữ liệu.

- Chọn báo cáo thiết kế, ấn chuột phải, chọn Insert Group Header/Footer.

- Chọn tên trường nhóm dữ liệu đưa vào đoạn Group Header.

Khi đó báo cáo được thiết kế như sau:

[missing_resource: .png]

Hình 13.5 Báo cáo có nhóm dữ liệu

Xem và xuất Data Report

Ta có thể xem thông tin và in báo cáo trên một cửa sổ riêng biệt sử dụng chế độ Print Preview bằng cách thi hành phương thức Show.

Khi đó báo cáo sẽ được hiển thị như sau:

[missing_resource: .png]

Hình 13.6: Thi hành báo cáo trong VB

Khi đó người sử dụng có thể duyệt qua các trang nếu như báo cáo có nhiều trang, cũng như chọn một trang báo cáo nào đó để in.

Ngoài ra người dùng có thể chọn Export báo cáo của mình ra tập tin có định dạng khác, các loại định dạng ở đây có thể là tập tin văn bản, tập tin HTML. Ta có thể chọn lựa xuất một số trang cụ thể nào đó hoặc toàn bộ báo cáo.

[missing_resource: .png]

Hình 13.7: Hộp thoại xuất báo cáo

SỬ DỤNG CRYSTAL REPORT ĐỂ LẬP BÁO CÁO

Crystal Report cho phép tạo báo cáo cơ sở dữ liệu trong ứng dụng viết bằng VB. Nó gồm 2 phần chủ yếu:

- Trình thiết kế báo cáo xác định dữ liệu sẽ đưa vào báo cáo và cách thể hiện của báo cáo.

- Một điều khiển ActiveX cho phép thi hành, hiển thị, điều khiển và in báo cáo khi thi hành ứng dụng.

Crystal Report không có sẵn khi cài VB6, ta cần cài đặt thêm. Chương trình cài đặt Crystal Report chỉ có trên bản Professional. Chạy tập tin Crystl32.exe trong thư mục \COMMON\TOOLS\VB\CRYSREPT.

Thiết kế báo cáo

Một điểm khác biệt khi dùng Crystal Report là ta không thiết lập báo cáo đi đôi với ứng dụng cụ thể. Ta sẽ xây dựng báo cáo trước và sau đó sẽ

gọi thi hành báo cáo từ phía ứng dụng, báo cáo không phải là một bộ phận thuộc ứng dụng. Cửa sổ thiết kế Crystal Report như hình bên dưới:

[missing_resource: .png]

Hình 13.8 Cửa sổ Crystal Report

Khi ta chọn tạo một báo cáo mới, Crystal Report trình bày một hộp thoại cho phép lựa chọn một trong nhiều những khuôn mẫu báo cáo đã định sẵn.

[missing_resource: .png]

Hình 13.9 Hộp thoại chọn các mẫu

Kiểu báo cáo	Mô tả
Standard	Báo cáo sắp xếp thông tin theo dòng và cột, cho phép nhóm dữ liệu.
Listing	Báo cáo là danh sách dữ liệu liên tục không có tổng kết hay trường tổng cộng..
Cross-Tab	Sắp xếp dữ liệu theo hai chiều.
Mail label	Báo cáo được thiết kế để in dữ liệu theo cột cho nhãn thư.

Summary	Báo cáo chỉ hiển thị thông tin tổng quát, không chứa dữ liệu chi tiết.
Graph	Báo cáo thể hiện dữ liệu một cách trực quan bằng biểu đồ
Top N	Báo cáo cho phép chỉ hiển thị một số mẩu tin được chọn
Drill Down	Báo cáo cho phép nhấn đúp chuột lên dữ liệu tổng quát để hiển thị dữ liệu chi tiết.
Another	Các báo cáo có khuôn mẫu do người dùng định nghĩa trước đó.

Chúng ta xét qua một ví dụ sử dụng Crystal Report để lập báo cáo

- Khởi động Crystal Report và chọn New, chọn kiểu báo cáo là Standard.

- Tiếp theo chọn Data File.

- Trong hộp thoại chọn tập tin cơ sở dữ liệu, ta chỉ đến một tập tin cơ sở dữ liệu, sau đó ấn nút Done. Ta sẽ thấy các bảng cũng như các quan hệ giữa các bảng được hiển thị.

[missing_resource: .png]

Hình 13.10 Hộp thoại quan hệ giữa các bảng

Quan hệ giữa các bảng đã được xác định ở mức thiết kế cơ sở dữ liệu nên ta không cần phải thay đổi những mối liên kết này.

- Nhấn nút Next qua bước tiếp theo, ta sẽ chọn những trường tham gia vào báo cáo.

- Bước kế tiếp ta chọn qua Tab Sort để thực hiện việc sắp xếp dữ liệu.

- Tab Style cho phép chọn các dạng khác nhau của báo cáo.

- Sau khi đã thiết kế xong, ta ấn Save để lưu lại báo cáo.

Khi mở lại báo cáo đã thiết kế, ta thấy Crystal Report hiển thị báo cáo ở hai mức, thiết kế và duyệt trước.

[missing_resource: .png]

Hình 13.11: Cửa sổ xem trước báo cáo và thiết kế báo cáo

Thi hành báo cáo trong ứng dụng thông qua điều khiển ActiveX của Crystal Report

Bước đầu tiên để có thể thi hành báo cáo Crystal Report, ta cần tham khảo đến điều khiển ActiveX của Crystal Report bằng cách thêm công cụ Crystal Report vào đề án của chúng ta.

Chọn công cụ Crystal Report và đưa vào ứng dụng, biểu tượng trên hộp công cụ như sau. Trong sự kiện Click của một nút lệnh, ta viết đoạn mã sau:

```
Private Sub Command1_Click()

CrystalReport1.ReportFileName = "d:\VB\bc.rpt"

CrystalReport1.PrintReport

End Sub
```

Thuộc tính ReportFileName xác định đường dẫn cũng như tên tập tin báo cáo.

Việc thi hành báo cáo được thực hiện nhờ vào thuộc tính PrintReport. Ngoài ra, báo cáo có thể thi hành bằng cách hiển thị trên một cửa sổ khác hoặc là xuất ra thẳng trên máy in, … Ta thiết đặt thuộc tính đó qua hộp thoại thuộc tính.

[missing_resource: .png]

Hình 13.11 Hộp thoại xác lập thuộc tính

Báo cáo thi hành trên một cửa sổ riêng biệt, ta có thể lựa chọn nhiều công việc như xem qua các trang, in ấn báo cáo, phóng to thu nhỏ …

[missing_resource: .png]

Hình 13.12: Báo cáo Crystal Report

LỜI KẾT

Chương Thiết lập báo cáo cũng là chương kết thúc của giáo trình Visual Basic. Tuy nhiên lập trình sự kiện và lập trình cơ sở dữ liệu với VB chỉ là một phần trong những khả năng mà VB mang lại. Hy vọng chúng tôi sẽ gặp lại bạn đọc trong những chuyên đề khác của VB.